



JBoss 企业级应用程序平台 6.3

管理和配置指南

适用于红帽 JBoss 企业版应用程序平台 6

JBoss 企业级应用程序平台 6.3 管理和配置指南

适用于红帽 JBoss 企业版应用程序平台 6

法律通告

Copyright © 2015 Red Hat, Inc.57.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

本书是安装和配置红帽 JBoss 企业版应用程序平台 6 及其补丁的指南。

目录

第 1 章 介绍	15
1.1. 关于 RED HAT JBOSS 企业版应用程序平台 6	15
1.2. JBOSS EAP 6 的功能	15
1.3. 关于 JBOSS EAP 6 操作模式	16
1.4. 关于独立服务器	16
1.5. 关于受管域	16
1.6. 关于域控制器	17
1.7. 关于域控制器发现和失效切换	17
1.8. 关于主机控制器	19
1.9. 关于服务器组	19
1.10. 关于 JBOSS EAP 6 配置集	20
第 2 章 应用服务器管理	21
2.1. 启动和停止 JBoss EAP 6	21
2.1.1. 启动 JBoss EAP 6	21
2.1.2. 将 JBoss EAP 6 作为独立服务器启动	21
2.1.3. 将 JBoss EAP 6 作为受管域启动	21
2.1.4. 配置受管域里主机的名字	22
2.1.5. 在两台主机上创建受管域	23
2.1.6. 用替代配置启动 JBoss EAP 6	24
2.1.7. 停止 JBoss EAP 6	26
2.1.8. 服务器启动参数和开关参考	28
2.2. 启动和停止服务器	30
2.2.1. 用管理 CLI 启动或停止服务器。	30
2.2.2. 使用管理控制台启动服务器	31
2.2.3. 使用管理控制台停止服务器	32
2.3. 文件系统路径	32
2.3.1. 文件系统路径	32
2.4. 配置文件	34
2.4.1. 关于 JBoss EAP 6 配置文件	34
2.4.2. 基于描述符的属性替换	35
2.4.3. 启用/禁用基于描述符的属性替换	36
2.4.4. 配置文件历史	38
2.4.5. 用以前的配置启动服务器	38
2.4.6. 使用管理 CLI 保存配置快照	39
2.4.7. 使用管理 CLI 加载配置快照	39
2.4.8. 使用管理 CLI 删除配置快照	40
2.4.9. 使用管理 CLI 列出所有的配置快照	41
第 3 章 管理接口	42
3.1. 管理应用服务器	42
3.2. 管理应用程序编程接口 (API)	42
3.3. 关于管理控制台和管理 CLI	43
3.4. 管理控制台	43
3.4.1. 管理控制台	43
3.4.2. 登录到管理控制台	43
3.4.3. 修改管理控制台的语言	44
3.4.4. EAP 控制台里的数据分析	45
3.4.5. 启用/禁用 EAP 控制台里的 Google Analytics	45
3.4.6. 使用管理控制台配置服务器	47
3.4.7. 在管理控制台里添加部署	48
3.4.8. 在管理控制台里创建新的服务器	48

3.4.9. 用管理控制台修改默认的日志级别	49
3.4.10. 在管理控制台里创建新的服务器组	49
3.5. 管理 CLI	50
3.5.1. 关于管理命令行接口 (Command Line Interface , CLI)	50
3.5.2. 启动管理 CLI	50
3.5.3. 退出管理 CLI	50
3.5.4. 用管理 CLI 连接受管服务器实例	51
3.5.5. 用管理 CLI 获取帮助	51
3.5.6. 以批模式使用管理 CLI	52
3.5.7. CLI 批处理模式命令	53
3.5.8. 在管理 CLI 里使用操作和命令	53
3.5.9. 管理 CLI 配置选项	56
3.5.10. 管理 CLI 命令参考	58
3.5.11. 管理 CLI 操作参考	59
3.6. 管理 CLI 操作	61
3.6.1. 用管理 CLI 显示资源属性	61
3.6.2. 在管理 CLI 里显示活动用户	63
3.6.3. 在管理 CLI 里显示系统和服务器信息	64
3.6.4. 用管理 CLI 显示操作描述	64
3.6.5. 用管理 CLI 显示操作名称	66
3.6.6. 用管理 CLI 显示可用资源	67
3.6.7. 用管理 CLI 显示可用资源的描述	72
3.6.8. 用管理 CLI 重载应用服务器	72
3.6.9. 用管理 CLI 关闭应用服务器	73
3.6.10. 使用管理 CLI 配置属性	74
3.6.11. 用管理 CLI 配置系统属性	75
3.7. 管理 CLI 命令历史	80
3.7.1. 使用管理 CLI 命令历史	80
3.7.2. 显示管理 CLI 命令历史	80
3.7.3. 清除管理 CLI 命令历史	80
3.7.4. 禁用管理 CLI 命令历史	81
3.7.5. 启用管理 CLI 命令历史	81
3.8. 管理接口审计日志	82
3.8.1. 关于管理接口审计日志	82
3.8.2. 从管理 CLI 里启用管理接口的审计日志	82
3.8.3. 关于管理接口的审计日志格式器	82
3.8.4. 关于管理接口的审计日志文件处理程序	83
3.8.5. 关于管理接口的审计日志 Syslog 处理程序	83
3.8.6. 启用发送到 Syslog 服务器的管理接口审计日志	84
3.8.7. 管理接口的审计日志选项	85
3.8.8. 管理接口的审计日志字段	85
第 4 章 用户管理	87
4.1. 用户创建	87
4.1.1. 为管理接口添加用户	87
4.1.2. 传入参数到用户管理 add-user 脚本	88
4.1.3. Add-user 命令行参数	88
4.1.4. 指定用户管理信息的替代属性文件	90
4.1.5. Add-user 脚本命令行示例	90
第 5 章 网络和端口配置	93
5.1. 接口	93
5.1.1. 关于接口	93

5.1.2. 配置接口	94
5.2. 套接字绑定组	97
5.2.1. 关于套接字绑定组	97
5.2.2. 配置套接字绑定	100
5.2.3. JBoss EAP 6 使用的网络端口	101
5.2.4. 关于套接字绑定组的端口偏移	104
5.2.5. 配置端口偏移	104
5.2.6. 配置 Remoting 里的消息大小	105
5.3. IPV6	105
5.3.1. 配置 IPV6 网络的 JVM Stack 首选项	105
5.3.2. 配置 IPV6 网络的接口声明	106
5.3.3. 配置 IPV6 地址的 JVM Stack 首选项	107
第 6 章 数据源管理	108
6.1. 介绍	108
6.1.1. 关于 JDBC	108
6.1.2. JBoss EAP 6 支持的数据库	108
6.1.3. 数据源的类型	108
6.1.4. 数据源示例	108
6.1.5. -ds.xml 文件的部署	109
6.2. JDBC 驱动	109
6.2.1. 用管理控制台安装 JDBC 驱动	109
6.2.2. 将 JDBC 驱动安装为核心模块	110
6.2.3. JDBC 驱动的下载位置	112
6.2.4. 访问供应商专有的类	112
6.3. NON-XA 数据源	113
6.3.1. 用管理界面创建一个 Non-XA 数据源	113
6.3.2. 用管理界面修改 Non-XA 数据源	115
6.3.3. 用管理界面删除 Non-XA 数据源	116
6.4. XA 数据源	117
6.4.1. 用管理界面创建 XA 数据源	117
6.4.2. 用管理界面修改 XA 数据源	118
6.4.3. 用管理界面删除 XA 数据源	119
6.4.4. XA Recovery	120
6.4.4.1. 关于 XA Recovery 模块	120
6.4.4.2. 配置 XA Recovery 模块	120
6.5. 数据源安全性	122
6.5.1. 关于数据源安全性	122
6.6. 数据源配置	123
6.6.1. 数据源参数	123
6.6.2. 数据源连接 URL	128
6.6.3. 数据源扩展	129
6.6.4. 查看数据源统计	130
6.6.5. 数据源统计	131
6.7. 数据源示例	132
6.7.1. PostgreSQL 数据源示例	132
6.7.2. PostgreSQL XA 数据源示例	133
6.7.3. MySQL 数据源示例	134
6.7.4. MySQL XA 数据源示例	135
6.7.5. Oracle 数据源示例	136
6.7.6. Oracle XA 数据源示例	137
6.7.7. Microsoft SQLServer 数据源示例	139
6.7.8. Microsoft SQLServer XA 数据源示例	140

6.7.9. IBM DB2 数据源示例	141
6.7.10. IBM DB2 XA 数据源示例	142
6.7.11. Sybase 数据源示例	143
6.7.12. Sybase XA 数据源示例	144
第 7 章 配置模块	146
7.1. 介绍	146
7.1.1. 模块	146
7.1.2. 全局模块	147
7.1.3. 模块的依赖关系	147
7.1.4. 子部署类加载器的隔离	147
7.2. 对所有的部署禁用子部署模块隔离 (SUB-DEPLOYMENT MODULE ISOLATION)	148
7.3. 添加模块到所有部署里	149
7.4. 定义外部 JBOSS MODULES 目录	150
7.5. 参考	150
7.5.1. 包括的模块	150
7.5.2. 动态模块命名	150
第 8 章 JSVC	152
8.1. 介绍	152
8.1.1. 关于 Jsvc	152
8.1.2. 用 Jsvc 启动和停止 JBoss EAP	152
第 9 章 全局 VALVE	157
9.1. 关于 VALVE	157
9.2. 关于全局 VALVE	157
9.3. 关于 AUTHENTICATOR VALVE	157
9.4. 安装 GLOBAL VALVE	157
9.5. 配置全局 VALVE	158
第 10 章 应用程序部署	160
10.1. 关于应用程序部署	160
10.2. 用管理控制台进行部署	161
10.2.1. 在管理控制台里管理应用程序的部署	161
10.2.2. 用管理控制台启用已部署的应用程序	161
10.2.3. 用管理控制台禁用已部署的应用程序	162
10.3. 用管理 CLI 进行部署	162
10.3.1. 在管理 CLI 里管理应用程序的部署	162
10.3.2. 用管理 CLI 在独立服务器里部署应用程序	163
10.3.3. 用管理 CLI 卸载独立服务器里的应用程序	163
10.3.4. 用管理 CLI 在受管域里部署应用程序	164
10.3.5. 用管理 CLI 卸载受管域里的应用程序	164
10.4. 用 HTTP API 进行部署	165
10.4.1. 用 HTTP API 部署应用程序	165
10.5. 用部署扫描器进行部署	168
10.5.1. 在部署扫描器 (Deployment Scanner) 里管理应用程序的部署	168
10.5.2. 用部署扫描器部署应用程序到独立服务器实例	168
10.5.3. 用部署扫描器卸载独立服务器实例的应用程序	169
10.5.4. 用部署扫描器在独立服务器实例里重新部署应用程序	170
10.5.5. 对部署扫描器 Marker 文件的引用	170
10.5.6. 对部署扫描器属性的引用	171
10.5.7. 配置部署扫描器	172
10.5.8. 用管理 CLI 配置部署扫描器	172
10.6. 用 MAVEN 进行部署	175

10.6.1. 用 Maven 管理应用程序部署	175
10.6.2. 用 Maven 部署应用程序	175
10.6.3. 用 Maven 卸载应用程序	177
10.7. 控制 JBOSS EAP 6 里部署应用程序的顺序	178
10.8. 部署描述符覆盖	179
第 11 章 保护 JBOSS EAP 6	180
11.1. 关于安全子系统	180
11.2. 关于安全子系统的结构	180
11.3. 配置安全子系统	181
11.4. 关于 DEEP COPY SUBJECT 模式	182
11.5. 启用 DEEP COPY SUBJECT 模式	182
11.6. 安全域	183
11.6.1. 关于安全域	183
11.6.2. 关于 Picketbox	183
11.6.3. 关于验证	184
11.6.4. 配置安全域的验证	184
11.6.5. 关于授权	185
11.6.6. 配置安全域里的授权	185
11.6.7. 关于安全性审计	187
11.6.8. 配置安全审计	187
11.6.9. 关于审计日志	188
11.6.10. 关于安全性映射	188
11.6.11. 在安全域里配置安全映射	188
11.6.12. 在应用程序里使用安全域	189
11.6.13. Java 容器授权合约 (JACC)	192
11.6.13.1. 关于 Java 容器授权合约 (JACC)	192
11.6.13.2. 配置 Java 容器授权合约 (JACC) 的安全性	192
11.6.14. Java 容器验证 SPI (JASPI)	193
11.6.14.1. 关于 Java 容器验证 SPI (JASPI) 的安全性	193
11.6.14.2. 配置 Java 容器验证 SPI (JASPI) 的安全性	193
11.7. 保证 IIOP 的安全	194
11.7.1. 关于 JBoss IIOP	194
11.7.2. 关于 IOR	194
11.7.3. IOR 安全参数	195
11.8. 管理接口的安全性	196
11.8.1. 默认的用户安全性配置	196
11.8.2. 高级管理接口配置概述	197
11.8.3. 关于 LDAP	198
11.8.4. 在管理接口里使用 LDAP 进行验证	198
11.8.5. 禁用 HTTP 管理接口	201
11.8.6. 从默认的安全区删除无提示验证	203
11.8.7. 禁用对 JMX 子系统的远程访问	204
11.8.8. 为管理接口配置安全区	205
11.9. 用基于角色的访问控制来保护管理接口	206
11.9.1. 关于基于角色的访问控制 (Role-Based Access Control , RBAC)	206
11.9.2. 管理控制台和 CLI 里基于角色的访问控制	206
11.9.3. 支持的验证模式	206
11.9.4. 标准角色	207
11.9.5. 关于角色权限	208
11.9.6. 关于约束	209
11.9.7. 关于 JMX 和基于角色的访问控制	210
11.9.8. 配置基于角色的访问控制	210

11.9.8.1. RBAC 配置任务概述	210
11.9.8.2. 启用基于角色的访问控制	210
11.9.8.3. 修改权限组合策略	212
11.9.9. 管理角色	213
11.9.9.1. 关于角色成员资格	213
11.9.9.2. 配置用户和角色的分配	213
11.9.9.3. 用 jboss-cli.sh 配置用户角色分配	216
11.9.9.4. 关于角色和用户组	219
11.9.9.5. 配置组角色的分配	220
11.9.9.6. 用 jboss-cli.sh 配置组角色	223
11.9.9.7. 关于用 LDAP 进行授权和组加载	226
username-to-dn	227
组搜索	228
通用组搜索	230
11.9.9.8. 关于带作用域的角色	232
11.9.9.9. 创建带作用域的角色	232
11.9.10. 配置约束	234
11.9.10.1. 配置 Sensitivity 约束	234
11.9.10.2. 配置应用程序资源约束	236
11.9.10.3. 配置 Vault 表达式约束	237
11.9.11. 约束引用	238
11.9.11.1. 应用程序资源约束引用	238
11.9.11.2. 安全约束引用	240
11.10. 网络安全性	248
11.10.1. 保护管理接口	248
11.10.2. 指定 JBoss EAP 6 使用的网络接口	248
11.10.3. JBoss EAP 6 使用的网络端口	249
11.10.4. 配置和 JBoss EAP 6 一起使用的网络防火墙	252
11.11. JAVA 安全性管理者	255
11.11.1. 关于 Java 安全性管理者	255
11.11.2. 在 Java 安全管理者里运行 JBoss EAP 6	255
11.11.3. 关于 Java 安全管理者策略	256
11.11.4. 编写 Java 安全性管理者策略	257
11.11.5. 调试安全管理者策略	258
11.12. SSL 加密	258
11.12.1. 对 JBoss EAP 6 Web 服务器实施 SSL 加密	258
11.12.2. 生成 SSL 密钥和证书	260
11.12.3. SSL 连接器引用	263
11.13. 用于敏感字符串的密码库	268
11.13.1. 关于保护明码文件里的敏感字符	268
11.13.2. 创建一个 Java 密钥库来存储敏感信息	268
11.13.3. 设置密钥库密码的掩码并初始化密码库	271
11.13.4. 配置 JBoss EAP 6 来使用密码库	271
11.13.5. 配置 JBoss EAP 6 来使用自定义的密码库实现	273
11.13.6. 在 Java 密钥库里保存和获取加密的敏感字符串	274
11.13.7. 存储和解析应用程序里的敏感字符串	276
11.14. FIPS 140-2 兼容加密	278
11.14.1. 关于 FIPS 140-2 兼容性	278
11.14.2. 兼容 FIPS 140-2 的密码	278
11.14.3. 在红帽企业版 Linux 6 上启用 SSL 的 FIPS 140-2 加密	279
11.14.4. 在 Apache HTTP 服务器里启用 FIPS 140-2 加密	282
第 12 章 安全管理引用	283

12.1. 包括的验证模块	283
12.2. 包括的授权模块	305
12.3. 所包括的安全映射模块	306
12.4. 包括的安全审计供应商模块	310
第 13 章 子系统配置	311
13.1. 子系统配置概述	311
第 14 章 日志子系统	312
14.1. 介绍	312
14.1.1. 日志概述	312
14.1.2. JBoss LogManager 支持的应用程序日志框架	312
14.1.3. 配置引导日志	312
14.1.4. 关于垃圾收集日志	313
14.1.5. 隐性的 Logging API 依赖关系	313
14.1.6. 默认的日志文件位置	313
14.1.7. 日志的过滤器表达式	314
14.1.8. 关于日志级别	315
14.1.9. 支持的日志级别	316
14.1.10. 关于日志类别	316
14.1.11. 关于 Root Logger	317
14.1.12. 关于日志处理程序	317
14.1.13. 日志处理程序的类型	317
14.1.14. 关于日志格式器	318
14.1.15. 日志格式器语法	318
14.2. 在管理控制台里配置日志	319
14.3. CLI 里的日志配置	320
14.3.1. 用 CLI 配置 Root Logger	320
14.3.2. 在 CLI 里配置日志类别	322
14.3.3. 在 CLI 里配置控制台日志处理程序	324
14.3.4. 在 CLI 里配置文件日志处理程序	327
14.3.5. 在 CLI 里配置定期日志处理程序	331
14.3.6. 在 CLI 里配置 Size 日志处理程序	335
14.3.7. 在 CLI 里配置 Async 日志处理程序	341
14.3.8. 配置 syslog 处理程序	344
14.3.9. 在 CLI 里配置自定义日志格式器	345
14.4. 重新部署日志子系统	346
14.4.1. 关于 Per-deployment 日志	346
14.4.2. 禁用 Per-deployment 日志	346
14.5. 日志配置集	347
14.5.1. 关于日志配置集	347
14.5.2. 用 CLI 创建新的日志配置集	347
14.5.3. 用 CLI 配置日志配置集	347
14.5.4. 指定应用程序里的日志配置集	348
14.5.5. 日志配置集配置示例	349
14.6. 日志配置属性	351
14.6.1. Root Logger 属性	351
14.6.2. 日志类别属性	351
14.6.3. Console Log Handler 的属性	351
14.6.4. 文件处理程序属性	352
14.6.5. 定期日志处理程序属性	353
14.6.6. Size 日志处理程序属性	354
14.6.7. 异步日志处理程序属性	355

14.7. 用于日志的 XML 配置示例	356
14.7.1. 用于 Root Logger 的 XML 配置示例	356
14.7.2. 用于日志类别的 XML 配置示例	356
14.7.3. 用于 Console Log Handler 的 XML 配置示例	356
14.7.4. 用于 File 日志处理程序的 XML 配置示例	356
14.7.5. 用于 Periodic 日志处理程序的 XML 配置示例	356
14.7.6. 用于 Size 日志处理程序的 XML 配置示例	357
14.7.7. 用于 Async 日志处理程序的 XML 配置示例	357
第 15 章 INFINISPAN	358
15.1. 关于 INFINISPAN	358
15.2. 群集模式	358
15.3. 缓存容器	359
15.4. 缓存库	360
15.5. 关于 INFINISPAN 统计	361
15.6. 启用 INFINISPAN 统计信息的采集	361
15.6.1. 在启动配置文件里启用 Infinispan 统计信息的采集	361
15.6.2. 从管理 CLI 启用 Infinispan 统计信息的采集	362
15.6.3. 检验是否已启用 Infinispan 统计信息的采集	362
15.7. JGROUPS	363
15.7.1. 关于 JGroups	363
第 16 章 JVM	364
16.1. 关于 JVM	364
16.1.1. 关于 JVM 设置	364
16.1.2. 在管理控制台里显示 JVM 状态	365
16.1.3. 配置 JVM	366
第 17 章 WEB 子系统	368
17.1. 配置 WEB 子系统	368
17.2. 替换默认的 WELCOME WEB 应用程序	372
第 18 章 WEB SERVICES 子系统	373
18.1. 配置 WEB SERVICES 选项	373
第 19 章 HTTP 群集和负载均衡	375
19.1. 介绍	375
19.1.1. 关于高可用性和负载均衡群集	375
19.1.2. 可从高可用性受益的组件	375
19.1.3. HTTP 连接器概述	376
19.1.4. 工作节点	377
19.2. 连接器配置	377
19.2.1. 为 JBoss EAP 6 里的 HTTP 连接器定义线程池	378
19.3. WEB SERVER 配置	380
19.3.1. 关于独立 Apache HTTP 服务器	381
19.3.2. 安装 JBoss EAP 6 附带的 Apache HTTP 服务器 (ZIP 方式)	381
19.3.3. 在 Red Hat 企业版 Linux 5/6/7 (RHEL) 里用 RPM 方式安装 Apache HTTP 服务器	383
19.3.4. httpd 上的 mod_cluster 配置	384
19.3.5. 将外部 Web 服务器用作 JBoss EAP 6 应用程序的 Web 前端	388
19.3.6. 配置 JBoss EAP 6 接受外部 Web 服务器的请求	389
19.4. 群集	390
19.4.1. Clustering 子系统使用 TCP 通讯	390
19.4.2. 配置 JGroups 子系统使用 TCP	391
19.4.3. 禁用 mod_cluster 子系统的广告	392

19.4.4. 为 HornetQ 群集切换 UDP 至 TCP	394
19.5. WEB、HTTP 连接器和 HTTP 群集	396
19.5.1. 关于 mod_cluster HTTP 连接器	396
19.5.2. 配置 mod_cluster 子系统	396
19.5.3. 安装 mod_cluster 模块至 Apache HTTP 服务器或 JBoss Enterprise Web Server (ZIP 方式)	409
19.5.4. 安装 mod_cluster 模块至 Apache HTTP 服务器或 JBoss Enterprise Web Server (RPM 方式)	411
19.5.5. 为启用 mod_cluster 的 Web 服务器配置服务器的 Advertisement 属性	412
19.5.6. 配置 mod_cluster 工作节点	414
19.5.7. 在群集间移植流量	419
19.6. APACHE MOD_JK	419
19.6.1. 关于 Apache mod_jk HTTP 连接器	419
19.6.2. 配置 JBoss EAP 6 用 Apache Mod_jk 进行通讯	420
19.6.3. 安装 mod_jk 模块到 Apache HTTP 服务器 (ZIP 方式)	420
19.6.4. 安装 Mod_jk 模块至 Apache HTTP Server (RPM)	424
19.6.5. Apache Mod_jk 工作节点的配置	427
19.7. APACHE MOD_PROXY	429
19.7.1. 关于 Apache mod_proxy HTTP 连接器	429
19.7.2. 安装 Mod_proxy HTTP 连接器到 Apache HTTP 服务器	429
19.8. MICROSOFT ISAPI	432
19.8.1. 关于 Internet Server API (ISAPI) HTTP 连接器	432
19.8.2. 下载并解压用于 Microsoft IIS 的 Webserver Connector Natives	432
19.8.3. 配置 Microsoft IIS 使用 ISAPI Redirector	432
19.8.4. 配置 ISAPI Redirector 发送客户请求到 JBoss EAP 6	434
19.8.5. 配置 ISAPI Redirector 在多个 JBoss EAP 6 服务器间平衡客户请求	436
19.9. ORACLE NSAPI	438
19.9.1. 关于 Netscape Server API (NSAPI) HTTP 连接器	439
19.9.2. 在 Oracle Solaris 上配置 NSAPI Connector	439
19.9.3. 将 NSAPI 配置为 Basic HTTP Connector	440
19.9.4. 配置 NSAPI 为负载均衡群集	442
第 20 章 MESSAGING	445
20.1. 介绍	445
20.1.1. HornetQ	445
20.1.2. 关于 Java 消息服务 (Java Messaging Service , JMS)	445
20.1.3. 支持的消息风格	445
20.2. 传输配置	445
20.2.1. 关于接收器 (Acceptor) 和连接器 (Connector)	446
20.2.2. 配置 Netty TCP	446
20.2.3. 配置 Netty Secure Sockets Layer (SSL)	448
20.2.4. 配置 Netty HTTP	449
20.2.5. 配置 Netty Servlet	450
20.3. 关于 JAVA 命名和目录接口 (JAVA NAMING AND DIRECTORY INTERFACE , JNDI)	452
20.4. 处理大型消息	452
20.4.1. 处理大型消息	452
20.4.2. 配置 HornetQ 大型消息	452
20.4.3. 配置参数	453
20.5. 分页	454
20.5.1. 关于分页	454
20.5.2. 分页文件	454
20.5.3. 配置分页文件夹	454
20.5.4. 分页模式	454
20.6. 配置	456
20.6.1. 配置 JMS 服务器	456

20.6.2. 配置 JMS 地址设置	461
20.6.3. 用 HornetQ 配置消息系统	464
20.6.4. 为 HornetQ 启用日志	464
20.6.5. 配置 HornetQ Core Bridge	465
20.6.6. 配置 JMS 桥	466
20.6.7. 配置延迟的重递送	468
20.6.8. 配置 Dead Letter 地址	468
20.6.9. 配置消息过期地址	469
20.6.10. 对 HornetQ 配置属性的引用	469
20.6.11. 设置消息过期	476
20.7. 消息分组	477
20.7.1. 关于消息分组	477
20.7.2. 在客户端使用 HornetQ Core API	477
20.7.3. 为 Java Messaging Service (JMS) 客户配置服务器	477
20.7.4. 群集分组	478
20.7.5. 群集分组的最佳实践	479
20.8. 重复消息的检测	479
20.8.1. 关于重复消息的检测	479
20.8.2. 对消息发送使用重复消息检测	479
20.8.3. 配置重复 ID 缓存	480
20.8.4. 对桥和群集连接使用重复消息检测	480
20.9. JMS 桥	481
20.9.1. 关于消息桥 (Bridge)	481
20.9.2. 创建 JMS 桥	481
20.10. 持久化	483
20.10.1. 关于 HornetQ 里的持久化	483
20.11. HORNETQ 群集	484
20.11.1. 关于服务器发现	485
20.11.2. 广播组	485
20.11.2.1. UDP 广播组	485
20.11.2.2. JGroups 广播组	486
20.11.3. 发现组	487
20.11.3.1. 配置服务器上的 UDP 发现组	487
20.11.3.2. 配置服务器上的 JGroups 发现组	488
20.11.3.3. 为 Java Messaging Service (JMS) 客户配置发现组	489
20.11.3.4. 为 Core API 配置发现	490
20.11.4. 服务器端的负载平衡	490
20.11.4.1. 配置群集连接	490
20.12. 高可用性	493
20.12.1. 高可用性简介	493
20.12.2. 关于 HornetQ 共享存储	494
20.12.3. 关于 HornetQ 存储配置	494
20.12.4. 关于 HornetQ 的日志类型	494
20.12.5. 用共享存储配置 HornetQ 的专有拓扑结构	495
20.12.6. HornetQ 消息复制	496
20.12.7. 配置 HornetQ 服务器的复制	496
20.12.8. 关于高可用性 (HA) 失效切换	497
20.12.9. HornetQ 备份服务器上的部署	498
第 21 章 事务子系统	499
21.1. 事务子系统的配置	499
21.1.1. 事务配置概述	499
21.1.2. 配置事务管理者	499

21.1.3. 用 JTA 事务配置您的数据源	502
21.1.4. 配置 XA 数据源	503
21.1.5. 关于事务日志消息	503
21.1.6. 为事务子系统配置日志	504
21.2. 事务管理	505
21.2.1. 浏览并管理事务	505
21.3. 事务引用	509
21.3.1. JBoss 事务的错误和异常	509
21.3.2. JTA 事务的限制	509
21.4. ORB 配置	509
21.4.1. 关于公共对象请求代理体系结构 (Common Object Request Broker Architecture , CORBA)	509
21.4.2. 为 JTS 事务配置 ORB	510
21.5. JDBC 对象库的支持	511
21.5.1. 事务的 JDBC 库	511
第 22 章 邮件子系统	513
22.1. 在邮件子系统里使用自定义传输	513
第 23 章 EJB	515
23.1. 介绍	515
23.1.1. EJB 概述	515
23.1.2. 用于管理员的 EJB 概述	515
23.1.3. Enterprise Bean	515
23.1.4. Session Beans	515
23.1.5. Message-Driven Bean	516
23.2. 配置 BEAN POOLS	516
23.2.1. Bean 池	516
23.2.2. 创建 Bean 池	516
23.2.3. 删除 Bean 池	517
23.2.4. 编辑 Bean 池	518
23.2.5. 为 Session 和 Message-Driven Bean 分配 Bean 池	519
23.3. 配置 EJB 线程池	520
23.3.1. EJB 线程池	520
23.3.2. 创建线程池	521
23.3.3. 删除线程池	522
23.3.4. 编辑线程池	523
23.4. 配置 SESSION BEAN	524
23.4.1. Session Bean 访问超时	524
23.4.2. 设置默认的 Session Bean 访问超时时间	524
23.5. 配置 MESSAGE-DRIVEN BEAN	526
23.5.1. 为 Message-Driven Bean 设置默认的资源适配器	526
23.6. 配置 EJB3 定时器服务	527
23.6.1. EJB3 定时器服务	527
23.6.2. 配置 EJB3 定时器服务	527
23.7. 配置 EJB 异步调用服务	527
23.7.1. EJB3 异步调用服务	527
23.7.2. 配置 EJB3 异步调用服务线程池	527
23.8. 配置 EJB3 远程调用服务	528
23.8.1. EJB3 远程服务	528
23.8.2. 配置 EJB3 远程服务	528
23.9. 配置 EJB 2.X ENTITY BEANS	528
23.9.1. EJB Entity Bean	528
23.9.2. Container-Managed Persistence (容器管理的持久化)	529

23.9.3. 启用 EJB 2.x 的容器管理持久化	529
23.9.4. 配置 EJB 2.x 的容器管理持久化	529
23.9.5. HiLo 密钥生成器使用的 CMP 子系统属性	531
第 24 章 JAVA 连接器架构 (JCA)	532
24.1. 介绍	532
24.1.1. 关于 Java EE Connector API (JCA)	532
24.1.2. JAVA 连接器架构 (JCA)	532
24.1.3. 资源适配器	532
24.2. 配置 JAVA 连接器架构 (JAVA CONNECTOR ARCHITECTURE, JCA) 子系统	533
24.3. 部署资源适配器	538
24.4. 配置已部署的资源适配器	539
24.5. 资源适配器描述符参考	544
24.6. 查看定义的连接统计	548
24.7. 资源适配器的统计信息	548
24.8. 部署 WEBSphere MQ 资源适配器	549
24.9. 安装 JBoss Active MQ 资源适配器	554
24.10. 配置通用的 JMS 资源适配器以用于第三方的 JMS 供应商	554
第 25 章 在 AMAZON EC2 上部署 JBoss EAP 6	559
25.1. 介绍	559
25.1.1. 关于 Amazon EC2	559
25.1.2. 关于 Amazon Machine Instance (AMI)	559
25.1.3. 关于 JBoss Cloud Access	559
25.1.4. JBoss Cloud Access 的功能	559
25.1.5. 支持的 Amazon EC2 实例类型	560
25.1.6. 受支持的 Red Hat AMI	560
25.2. 在 AMAZON EC2 上部署 JBoss EAP 6	560
25.2.1. 在 Amazon EC2 上部署 JBoss EAP 6	560
25.2.2. 非群集的 JBoss EAP 6	561
25.2.2.1. 关于非群集的实例	561
25.2.2.2. 非群集实例	561
25.2.2.2.1. 启动非群集的 JBoss EAP 6 实例	561
25.2.2.2.2. 在非群集 JBoss EAP 6 实例上部署应用程序	562
25.2.2.2.3. 测试非群集 JBoss EAP 6 实例	564
25.2.2.3. 非群集的受管域	564
25.2.2.3.1. 启动一个实例作为域控制器	564
25.2.2.3.2. 启动一个或多个实例来充当主机控制器	567
25.2.2.3.3. 测试非群集 JBoss EAP 6 受管域	568
25.2.2.3.4. 配置 Amazon EC2 上的域控制器发现和失效切换	569
25.2.3. 群集的 JBoss EAP 6	570
25.2.3.1. 关于群集实例	571
25.2.3.2. 创建一个关系型数据库服务的数据库实例	571
25.2.3.3. 关于虚拟私有云	572
25.2.3.4. 创建虚拟私有云 (VPC)	572
25.2.3.5. 启动 Apache HTTP 服务器实例作为 VPC 的 mod_cluster 代理和 NAT 实例	573
25.2.3.6. 配置 VPC 私有子网的默认路由	575
25.2.3.7. 关于标识符和访问管理 (Identity and Access Management, IAM)	575
25.2.3.8. 配置 IAM 设置	575
25.2.3.9. 关于 S3 Bucket	576
25.2.3.10. 配置 S3 Bucket	576
25.2.3.11. 群集实例	578
25.2.3.11.1. 启动群集 JBoss EAP 6 AMI	578

25.2.3.11.2. 测试群集 JBoss EAP 6 实例	581
25.2.3.12. 群集的受管域	581
25.2.3.12.1. 启动一个实例作为群集域控制器	581
25.2.3.12.2. 启动一个或多个实例来充当群集主机控制器	584
25.2.3.12.3. 测试群集 JBoss EAP 6 受管域	586
25.3. 用 JBOSS OPERATIONS NETWORK (JON) 建立监控	587
25.3.1. 关于 AMI 监控	587
25.3.2. 关于连接性的要求	588
25.3.3. 关于网络地址转换 (Network Address Translation , NAT)	588
25.3.4. 关于 Amazon EC2 和 DNS	588
25.3.5. 关于 EC2 里的路由	589
25.3.6. 关于终止和重启 JON	589
25.3.7. 配置实例来注册 JBoss Operations Network	589
25.4. 用户脚本配置	590
25.4.1. 永久性的配置参数	590
25.4.2. 自定义脚本参数	593
25.5. 故障排除	594
25.5.1. 关于 Amazon EC2 的故障解除	594
25.5.2. 诊断信息	594
附录 A. 补充的引用	595
A.1. 从红帽 CUSTOMER PORTAL 下载文件	595
A.2. 在红帽企业版 LINUX 上配置默认的 JDK	595
附录 B. 修订记录	597

第 1 章 介绍

1.1. 关于 RED HAT JBOSS 企业版应用程序平台 6

Red Hat JBoss 企业版应用程序平台 6 (JBoss EAP 6) 是一个构建在开放标准上并和 Java EE 6 规格兼容的中间件平台。它集成了 JBoss Application Server 7 和高可用性的群集、消息系统、分布式缓存以及其他技术。

JBoss EAP 6 使用了新的模块化结构，允许在有需要时才启用服务，从而提高了启动速度。

管理控制台和管理命令行界面使您不需要再编辑 XML 配置文件并增添了使用脚本和自动化任务的能力。

此外，JBoss EAP 6 包含了 API 和开放框架以用于快速开发安全和可扩充的 Java EE 应用程序。

[提交 bug 报告](#)

1.2. JBOSS EAP 6 的功能

表 1.1. 6.3.0 版本的功能

功能	描述
Java 认证	认证了 Java EE 6 的 Full 和 Web 配置集。
受管域	<ul style="list-style-type: none"> 对多个服务器实例和物理主机的集中管理，但独立服务器也允许单个的服务器实例。 对每个服务器组的配置、部署、套接字绑定、模块、扩展和系统属性的管理。 对应用程序安全性（包括安全域）的集中式和简化管理。
管理控制台和管理 CLI	新的域或独立服务器管理界面。您不再需要编辑 XML 配置文件了。管理 CLI 也包括可以用脚本完成和管理任务自动化的批模式。
简化的目录格式	modules 目录现在包括了所有的应用服务器模块。公用的和服务器专有的 lib 目录已被抛弃。 domain 和 standalone 目录包含了域和独立部署的 artifact 和配置文件。
模块化的类加载机制	模块将根据需要来加载和卸载。这将提高性能和增强安全性、以及更快的启动和重启速度。
流线型的数据源管理	数据库驱动可以像其他服务一样部署。此外，数据源可以在管理控制台和管理 CLI 里直接创建和管理。
资源使用更少的且效率更高。	JBoss EAP 6 使用了更少的系统资源并比以前的版本更高效地利用资源。JBoss EAP 6 的启动与停止也比 JBoss EAP 5 快。

[提交 bug 报告](#)

1.3. 关于 JBoss EAP 6 操作模式

JBoss EAP 6 为 JBoss EAP 6 实例提供两种操作模式：独立服务器（**standalone server**）和受管域（**managed domain**）。

这两种模式的区别在于如何管理服务器，而不是它们为满足最终用户请求而提供的功能。请注意，高可用性（HA）群集功能可以通过这两种模式来实现。您可配置一组独立服务器来组成 HA 群集。

[提交 bug 报告](#)

1.4. 关于独立服务器

独立服务器以独立的进程运行，它和以前的 JBoss EAP 版本的唯一运行模式类似。

作为独立服务器运行的 JBoss EAP 实例只是一个单一实例，但也可以运行在群集配置里。

[提交 bug 报告](#)

1.5. 关于受管域

受管域是通过单个控制点来管理多个 JBoss EAP 6 实例的模式。

集中管理的 JBoss EAP 6 服务器集合被称作域的成员。域里所有的 JBoss EAP 6 实例共享一个公共的管理策略。

域由一个域控制器、一个或多个主机控制器以及对应每台主机的零或多个服务器组组成。

域控制器是控制域的中心点。它确保每个服务器都按照域的管理策略进行配置。域控制器也是主机控制器。

主机控制器是一个运行 **domain.sh** 或 **domain.bat** 脚本的物理或虚拟主机。我们配置主机控制器将域管理任务委托给域控制器。

每台主机上的主机控制器和域控制器进行交互以控制运行在主机上的应用服务器实例的生命周期，并协助域控制器管理主机。每台主机都可以包含多个服务器组。

服务器组是一系列安装了 JBoss EAP 6 的服务器实例并作为一个服务器进行管理和配置。域控制器管理部署至服务器组的应用程序的配置。因此，服务器组里的每个服务器都共享相同的配置和部署。

域控制器、单个主机控制器和多个服务器是可以运行在相同物理系统上的相同 JBoss EAP 6 实例里的。

主机控制器捆绑在专门的物理（或虚拟）主机上。如果使用不同的配置，您可以在相同的硬件上运行多个主机控制器，确保端口和其他资源不会冲突。

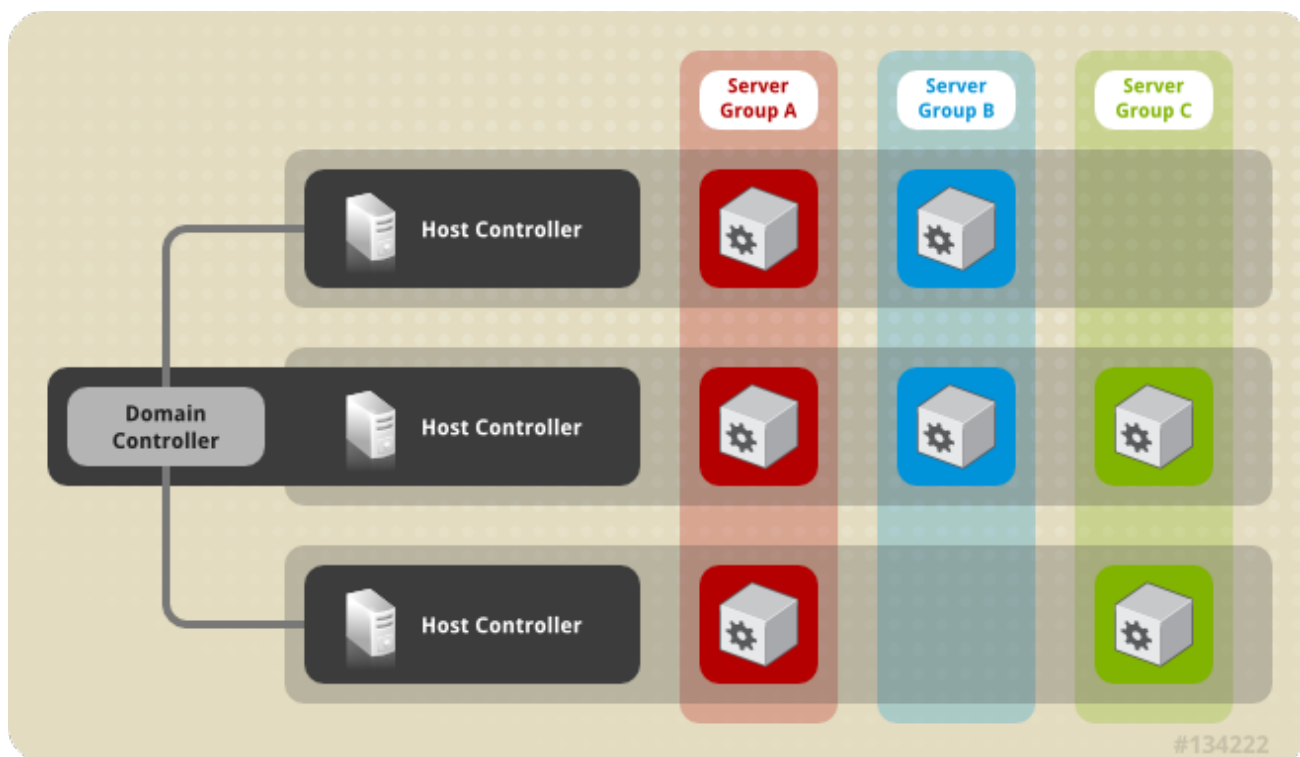


图 1.1. 受管域的图形表示形式

[提交 bug 报告](#)

1.6. 关于域控制器

域控制器是一个充当域集中管理点的 JBoss EAP 6 服务器实例。您可以配置主机控制器为域控制器。域控制器的主要职责是：

域控制器的主要职责是：

- 维护域的集中管理策略。
- 确保所有的主机控制器都意识到其当前的内容。
- 协助主机控制器以确保所有运行的 JBoss EAP 6 实例都按照这个策略进行配置。

集中管理策略默认保存在 `domain/configuration/domain.xml` 文件里，它位于域控制器的主机文件系统中解压的 JBoss EAP 6 安装目录里。

`domain.xml` 必须位于作为域控制器运行的主机控制器的 `domain/configuration/` 目录里。这个文件对于不是作为域控制器的主机控制器上的安装来说并不是强制的，虽然 `domain.xml` 文件的出现也没有害处。

`domain.xml` 文件包含不同配置集的配置，它们可都用于运行在域里的服务器实例。配置集的配置包含组成配置集的不同子系统的详细配置。域配置也包含对套接字组和服务器组的定义。

[提交 bug 报告](#)

1.7. 关于域控制器发现和失效切换

设置受管域时，每个主机控制器都必须用联系域控制器所需的信息进行配置。在 JBoss EAP 6.3 里，每个主机控制器都可以配置多个选项来寻找域控制器。主机控制器轮询选项列表，直至成功找到域控制器。

这个允许主机控制器用联系信息重新配置为备份域控制器。如果主域控制器出现问题，备份主机控制器可以提升为主域控制器，提升后它可以自动失效切换至新的主控制器。

下面是一个如何用多个选项寻找域控制器来配置主机控制器的例子。

```
<domain-controller>
  <remote security-realm="ManagementRealm">
    <discovery-options>
      <static-discovery name="primary" host="172.16.81.100"
port="9999"/>
      <static-discovery name="backup" host="172.16.81.101"
port="9999"/>
    </discovery-options>
  </remote>
</domain-controller>
```

静态的发现选项包括下列强制的属性：

名称

这个域控制器发现选项的名称

主机

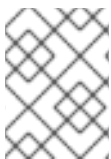
远程域控制器的主机名称。

重点

远程域控制器的端口。

在上面的例子里，第一个发现选项是期望成功的选项。第二个可以用在失效切换的情况下。

如果主域控制器出现问题，用 **--backup** 选项启动的主机控制器将被提升为域控制器。



注意

用 **--backup** 选项启动主机控制器将导致控制器维护域配置的一个本地拷贝。如果主机控制器被重设为域控制器，这个配置将被使用。

过程 1.1. 将主机控制器提升为域控制器

1. 确保原来的域控制器已停止。
2. 使用管理 CLI 连接至将成为新域控制器的主机控制器。
3. 执行下列命令来将主机控制器配置为新的域控制器。

```
/host=HOST_NAME:write-local-domain-controller
```

4. 执行下列命令来重新加载主机控制器。

```
reload --host=HOST_NAME
```

步骤 2 里选择的主机控制器将充当域控制器。

[提交 bug 报告](#)

1.8. 关于主机控制器

当 `domain.sh` 或 `domain.bat` 脚本在主机上运行时启动主机控制器。

主机控制器的主要职责是服务器管理。它委托域管理任务并负责启动和停止运行在主机上的单独应用服务器进程。

它和域控制器进行交互以帮助管理服务器和域控制器间的通讯。域里多个主机服务器可以和单个域控制器进行交互。因此，所有的主机控制器和运行在单一域模式下的服务器实例可以拥有单一的域控制器且必须属于相同的域。

每个主机控制器默认都从主机的文件系统中解压的 JBoss EAP 6 安装的 `domain/configuration/host.xml` 文件读取配置。`host.xml` 包含下列特定主机专有的配置信息：

- 要从这个安装位置运行的 JBoss EAP 6 实例名称
- 任何下列的配置：
 - 主机控制器如何联系域控制器来注册自身并访问域配置。
 - 如何找到并联系远程的域控制器。
 - 主机控制器将充当域控制器
- 这些条目是本地物理安装专有的配置。例如，`domain.xml` 里声明的命名接口定义可以映射到 `domain.xml` 里实际的主机专有的 IP 地址。`domain.xml` 里的绝对路径名可以映射到 `host.xml` 里实际的文件系统路径。

[提交 bug 报告](#)

1.9. 关于服务器组

服务器组是一个进行集中管理和配置的服务器实例集合。在受管域里，每个应用服务器实例都属于一个服务器组，即使它只是唯一的成员。组里的服务器实例共享相同的配置集和部署的内容。

域控制器和主机控制器在域里的每个服务器组的所有服务器实例上强制实施标准的配置。

域可由多个服务器组组成。不同的服务器组可以用不同的配置集和部署来配置。域可以用提供不同服务的不同服务器层来配置。

不同的服务器组也可以由相同的配置集和部署。例如，允许应用程序升级的轮换，当应用程序在某个服务器组上升级后再在另外的服务器组里更新，从而避免了整个服务的中断。

下面是一个服务器组定义示例：

```
<server-group name="main-server-group" profile="default">
  <socket-binding-group ref="standard-sockets"/>
  <deployments>
    <deployment name="foo.war_v1" runtime-name="foo.war"/>
    <deployment name="bar.ear" runtime-name="bar.ear"/>
  </deployments>
</server-group>
```

服务器组包括下列强制的属性：

- **name**：服务器组的名称
- **profile**：服务器组的配置集的名称
- **socket-binding-group**：用于组里服务器的默认套接字绑定组的名称。在 **host.xml** 里可以对每个服务器覆盖这个名称。然而，对于每个服务器组这都是一个强制性的元素，如果缺失，域将无法启动。

服务器组包含下列可选属性：

- **deployments**：部署在组里服务器上的部署内容
- **system-properties**：组里服务器上设置的系统属性
- **jvm**：组里所有服务器的默认 JVM。主机控制器将这些设置和 **host.xml** 里的其他配置进行合并以生成启动服务器的 JVM 的设置。

[提交 bug 报告](#)

1.10. 关于 JBOSS EAP 6 配置集

以前 JBoss EAP 版本里使用的配置集的概念不再使用了。JBoss EAP 6 现在使用更少的配置文件来保存配置信息。

模块和驱动现在是根据需要加载的，所以以前 JBoss EAP 6 版本里让服务器更高效地启动的 **default** 配置集的概念不再适用。

在部署期间，模块依赖关系将被确定、排序且由服务器或域控制器进行解析，并按正确的顺序加载。当没有部署需要模块时，它将被卸载。

您可以从配置里删除子系统来禁用模块或手动卸载驱动或其他服务。然而，大多数情况下这是不必要的。如果你的应用程序没有使用模块，它不会被加载。

[提交 bug 报告](#)

第 2 章 应用服务器管理

2.1. 启动和停止 JBOSS EAP 6

2.1.1. 启动 JBoss EAP 6

以下列方式之一启动 JBoss EAP 6：

- [第 2.1.2 节“将 JBoss EAP 6 作为独立服务器启动”](#)
- [第 2.1.3 节“将 JBoss EAP 6 作为受管域启动”](#)

[提交 bug 报告](#)

2.1.2. 将 JBoss EAP 6 作为独立服务器启动

介绍

本节涵盖将 JBoss EAP 6 作为独立服务器启动的步骤

过程 2.1. 将平台服务作为独立服务器启动

1. 对于红帽企业版 Linux。
运行命令：`EAP_HOME/bin/standalone.sh`
2. 对于 Microsoft Windows 服务器。
运行命令：`EAP_HOME\bin\standalone.bat`
3. 可选：指定其他的参数。
要查看传入启动脚本的其他参数，请使用 `-h` 参数。

结果

JBoss EAP 6 服务器实例已启动。

[提交 bug 报告](#)

2.1.3. 将 JBoss EAP 6 作为受管域启动

操作顺序

域控制器必须在域里任何服务器组里的任何从服务器之前启动。先在域控制器上，然后在每个关联的主机控制器和其他主机上使用这个过程。

过程 2.2. 将平台服务作为受管域启动

1. 对于红帽企业版 Linux。
运行命令：`EAP_HOME/bin/domain.sh`
2. 对于 Microsoft Windows 服务器。
运行：`EAP_HOME\bin\domain.bat`
3. 可选：传递其他参数到启动脚本里。
请使用 `-h` 参数来获取传递到启动脚本里的参数列表。

结果

JBoss EAP 6 受管域实例已启动。

[提交 bug 报告](#)

2.1.4. 配置受管域里主机的名字

概述

运行在受管域里的每个主机都必须有一个唯一的名称。为了简化管理并允许在多个主机上使用相同的主机配置文件，服务器将使用下列次序来确定主机名。

1. `host.xml` 配置文件里的 `host` 元素的 `name` 属性，如果指定了的话。
2. `jboss.host.name` 系统属性的值。
3. `jboss.qualified.host.name` 系统属性里最后一个句点 (".") 后的值，如果没有句点则是整个值。
4. 基于 POSIX 的操作系统的 `HOSTNAME` 环境变量或 Microsoft Windows 的 `COMPUTERNAME` 环境变量里的句点 (".") 后的值，如果没有句点则是整个值。

关于设置环境变量的信息，请参考操作系统的相关文档。关于如何设置系统属性的信息，请参考第 3.6.11 节“用管理 CLI 配置系统属性”。

本节描述了如何通过系统属性或硬编码在配置文件里设置主机名。

过程 2.3. 使用系统属性配置主机名

1. 打开主机配置文件，如 `host.xml`。
2. 找到 `host` 元素，例如：

```
<host name="master" xmlns="urn:jboss:domain:1.6">
```

3. 如果有这些内容，请删除 `name="HOST_NAME"` 属性声明。`host` 元素应该类似于下面的例子。

```
<host xmlns="urn:jboss:domain:1.6">
```

4. 用 `-Djboss.host.name` 参数启动服务器，例如：

```
-Djboss.host.name=HOST_NAME
```

过程 2.4. 使用专有名称配置主机名

1. 用下列语法启动 JBoss EAP 从主机：

```
bin/domain.sh --host-config=HOST_FILE_NAME
```

例如：

```
bin/domain.sh --host-config=host-slave01.xml
```

2. 启动管理 CLI。

3. 使用下列语法来替代主机名：

```
/host=EXISTING_HOST_NAME:write-attribute(name="name",value=UNIQUE_HOST_NAME)
```

例如：

```
/host=master:write-attribute(name="name",value="host-slave01")
```

你应该看到下面的结果。

```
"outcome" => "success"
```

这修改了 **host-slave01.xml** 文件里的 **host name** 属性：

```
<host name="host-slave01" xmlns="urn:jboss:domain:1.6">
```

4. 您必须重新加载使用旧的主机名的服务器配置以完成这个过程。

```
reload --host=EXISTING_HOST_NAME
```

例如：

```
reload --host=master
```

[提交 bug 报告](#)

2.1.5. 在两台主机上创建受管域



注意

您可能需要配置防火墙以运行这个例子。

您可以在两台主机上创建受管域，其中一台主机是域控制器而另外一个普通主机。详情请参考 [第 1.6 节“关于域控制器”](#)。

- IP1 = IP address of the domain controller (Machine 1)
- IP2 = IP address of the host (Machine 2)

过程 2.5. 在两台主机上创建受管域

1. 在 Machine 1 上

- a. 使用 **add-user.sh** 脚本添加管理用户，例如 **slave01**，让主机可以通过域控制器进行验证。请注意 **add-user** 输出里的 **SECRET_VALUE**。
- b. 用 **host-master.xml** 配置文件启动域，它为指定的域控制器进行了预先配置。
- c. 用 **-bmanagement=\$IP1** 使域控制器对于其他主机可见。

—

```
[$JBASS_HOME/bin]$ ./domain.sh --host-config=host-master.xml -
bmanagement=$IP1
```

2. 在 Machine 2 上

- a. 用用户凭证更新 **\$JBASS_HOME/domain/configuration/host-slave.xml** 文件。

```
<?xml version='1.0' encoding='UTF-8'?>
  <host xmlns="urn:jboss:domain:1.6" name="slave01">
    <!-- add user name here -->
    <management>
      <security-realms>
        <security-realm name="ManagementRealm">
          <server-identities>
            <secret value="$SECRET_VALUE" />
            <!-- use secret value from add-user.sh
output-->
          </server-identities>
          ...
```

- b. 启动主机。

```
[$JBASS_HOME/bin]$ ./domain.sh --host-config=host-slave.xml -
Djboss.domain.master.address=$IP1 -b=$IP2
```

3. 现在我们可以管理域了。

通过 CLI：

```
[$JBASS_HOME/bin]$ ./jboss-cli.sh -c --controller=$IP1
```

通过 Web 控制台：

```
http://$IP1:9990
```

访问服务器的索引页：

```
http://$IP2:8080/
http://$IP2:8230/
```

[提交 bug 报告](#)

2.1.6. 用替代配置启动 JBoss EAP 6

如果您没有指定配置文件，服务器将用默认的文件启动。然而，当您启动服务器时，您可以手动指定一个配置文件。启动过程会稍有不同，这取决于您是使用受管域还是独立服务器、以及您使用的操作系统。

必须具备的条件

- 在使用替代配置文件之前，请将 **default** 配置作为模版使用。对于受管域，配置文件必须位于 **EAP_HOME/domain/configuration/** 目录。对于独立服务器，配置文件必须位于 **EAP_HOME/standalone/configuration/** 目录。



注意

EAP_HOME/docs/examples/configs/ 目录里包含了几个配置示例。请用这些例子来启用额外的功能，如群集或 Transactions XTS API。

过程 2.6. 用其他配置启动实例

1. 独立服务器

对于独立服务器，请将配置文件的名称作为 **--server-config** 参数的选项。配置文件必须位于 **EAP_HOME/standalone/configuration/** 目录里，而且您需要指定相对这个目录的路径。

例 2.1. 在红帽企业版 Linux 里对独立服务器使用其他的配置文件

```
[user@host bin]$ ./standalone.sh --server-config=standalone-
alternate.xml
```

这个例子使用了 **EAP_HOME/standalone/configuration/standalone-alternate.xml** 配置文件。

例 2.2. 在 Microsoft Windows 服务器里对独立服务器使用其他的配置文件

```
C:\EAP_HOME\bin> standalone.bat --server-config=standalone-
alternate.xml
```

这个例子使用了 **EAP_HOME\standalone\configuration\standalone-alternative.xml** 配置文件。

2. 受管域

对于受管域，请为 **--domain-config** 参数提供配置文件的名称。这个文件必须位于 **EAP_HOME/domain/configuration/** 目录，且您需要指定相对这个目录的路径。

例 2.3. 在红帽企业版 Linux 里对受管域使用其他的配置文件

```
[user@host bin]$ ./domain.sh --domain-config=domain-alternate.xml
```

这个例子使用了 **EAP_HOME/domain/configuration/domain-alternate.xml** 配置文件。

例 2.4. 在 Microsoft Windows 服务器里对受管域使用其他的配置文件

```
C:\EAP_HOME\bin> domain.bat --domain-config=domain-alternate.xml
```

这个例子使用了 **EAP_HOME\domain\configuration\domain-alternate.xml** 配置文件。

结果

使用替代配置运行了 JBoss EAP 6。

[提交 bug 报告](#)

2.1.7. 停止 JBoss EAP 6

您停止 JBoss EAP 6 的方式取决于它是如何启动的。本节涵盖停止交互式启动的实例、停止作为服务启动的实例以及停止用脚本复制至后台进程的实例。



注意

关于在受管域里停止服务器或服务器组的信息，请参考 [第 2.2.3 节“使用管理控制台停止服务器”](#)。关于使用管理 CLI 停止服务器的信息，请参考 [第 2.2.1 节“用管理 CLI 启动或停止服务器。”](#)。

- **过程 2.7. 停止 JBoss EAP 6 的实例**

- **停止从命令提示交互式启动的实例。**
在 JBoss EAP 6 运行的终端窗口里按 **Ctrl-C**。

- **过程 2.8. 停止作为操作系统服务启动的实例。**

根据操作系统使用下列步骤。

- ■ **Red Hat Enterprise Linux**

对于红帽企业版 Linux，如果您已经编写了服务脚本，请使用它的 **stop** 功能。这需要编写到脚本里。然后您可以使用 **service scriptname stop**，这里的 *scriptname* 是脚本名称。

- **Microsoft Windows Server**

在 Microsoft Windows 里，使用 **net service** 命令，或者通过控制面板里的 **Services** 小程序来停止服务。

- **过程 2.9. 停止在后台运行的示例（红帽企业版 Linux）**

1. 获取进程的 ID (PID)：

- **如果只有一个实例在运行（独立模式）**

下面的命令都会返回单个 JBoss EAP 6 实例的 PID：

- **pidof java**

- **jps**

（**jps** 命令将返回两个进程的 ID：**jboss-modules.jar** 及 **jps** 自身。请使用 **jboss-modules.jar** 的 ID 来停止 EAP 实例）

- **如果有多个 EAP 实例在运行（域模式）**

有多个 EAP 实例运行时要确定正确的进程需要使用更复杂的命令。

- **jps** 命令可以使用冗余模式来提供 java 进程的更多信息。

下面是确定不同 EAP 进程的冗余 **jps** 命令的摘要，包括 PID 和角色：

```

$ jps -v
12155 jboss-modules.jar -D[Server:server-one] -
XX:PermSize=256m -XX:MaxPermSize=256m -Xms1303m
...

12196 jboss-modules.jar -D[Server:server-two] -
XX:PermSize=256m -XX:MaxPermSize=256m -Xms1303m
...

12096 jboss-modules.jar -D[Host Controller] -Xms64m -
Xmx512m -XX:MaxPermSize=256m
...

11872 Main -Xms128m -Xmx750m -XX:MaxPermSize=350m -
XX:ReservedCodeCacheSize=96m -XX:+UseCodeCacheFlushing
...

11248 jboss-modules.jar -D[Standalone] -
XX:+UseCompressedOops -verbose:gc
...

12892 Jps
...

12080 jboss-modules.jar -D[Process Controller] -Xms64m -
Xmx512m -XX:MaxPermSize=256m
...

```

- **ps aux** 命令也可以用来返回多个 EAP 实例的信息。

下面是确定不同 EAP 进程的冗余 **ps aux** 命令的摘要，包括 PID 和角色：

```

$ ps aux | grep java
username 12080 0.1 0.9 3606588 36772 pts/0 Sl+ 10:09
0:01 /path/to/java -D[Process Controller] -server -Xms128m
-Xmx128m -XX:MaxPermSize=256m
...

username 12096 1.0 4.1 3741304 158452 pts/0 Sl+ 10:09
0:13 /path/to/java -D[Host Controller] -Xms128m -Xmx128m -
XX:MaxPermSize=256m
...

username 12155 1.7 8.9 4741800 344224 pts/0 Sl+ 10:09
0:22 /path/to/java -D[Server:server-one] -XX:PermSize=256m
-XX:MaxPermSize=256m -Xms1000m -Xmx1000m -server -
...

username 12196 1.8 9.4 4739612 364436 pts/0 Sl+ 10:09
0:22 /path/to/java -D[Server:server-two] -XX:PermSize=256m
-XX:MaxPermSize=256m -Xms1000m -Xmx1000m -server
...

```

在上面的例子里，**Process Controller** 进程是停止以停止整个域的进程。

grep 工具可以和这些命令一起使用来确定 **Process Controller** :

```
jps -v | grep "Process Controller"

ps aux | grep "Process Controller"
```

2. 运行 **kill PID** 给进程发送 **TERM** 信号，这里的 **PID** 是上面的命令确定的进程号。

结果

每个方法都可以干净地关闭 JBoss EAP 6，所以不会丢失数据。

[提交 bug 报告](#)

2.1.8. 服务器启动参数和开关参考

应用服务器启动脚本在运行时接受其他参数和开关。这些参数允许服务器根据 **standalone.xml**、**domain.xml** 和 **host.xml** 配置文件里定义的其他配置启动。这可能包括用其他套接字绑定集或次级配置启动服务器。在启动时使用 **help** 开关可以参考这些可用的参数列表。

例 2.5.

下面的例子和 [第 2.1.2 节 “将 JBoss EAP 6 作为独立服务器启动”](#) 和 [第 2.1.3 节 “将 JBoss EAP 6 作为受管域启动”](#) 里解释的服务器启动类似，但添加了 **-h** 或 **--help** 开关。下表解释了 **help** 开关的结果。

独立模式：

```
[localhost bin]$ standalone.sh -h
```

域模式：

```
[localhost bin]$ domain.sh -h
```

表 2.1. 运行时开关和参数表

参数或开关	模式	描述
--admin-only	独立服务器模式	设置服务器的运行类型为 ADMIN_ONLY 。这将导致它打开管理接口并接受管理请求，但不会启动其他运行时服务或接受最终用户请求。
--admin-only	域模式	设置主机控制器的运行类型为 ADMIN_ONLY 。这将导致它打开管理接口并接受管理请求，但不会启动其他运行时服务；如果这个主机控制器不是域的主控制器，将接受来自从主机控制器的转入连接。
-b <value> , -b=<value>	独立服务器模式，域模式	设置系统属性 jboss.bind.address 为给定的值。

参数或开关	模式	描述
-b<interface>=<value>	独立服务器模式，域模式	设置系统属性 jboss.bind.address.<interface> 为给定的值。
--backup	域模式	保持持久性域配置的备份，即使这个主机不是域控制器。
-c <config> , -c=<config>	独立服务器模式	要使用的服务器配置文件的名称。默认是 standalone.xml 。
-c <config> , -c=<config>	域模式	要使用的服务器配置文件的名称。默认是 domain.xml 。
--cached-dc	域模式	如果主机不是域控制器且在引导时无法联系域控制器，它将使用域配置的本地缓存备份。
--debug [<port>]	独立服务器模式	激活调试模式并用可选参数来指定端口。只有启动脚本支持才可以使用。
-D<name>[=<value>]	独立服务器模式，域模式	设置系统属性。
--domain-config=<config>	域模式	要使用的服务器配置文件的名称。默认是 domain.xml 。
-h , --help	独立服务器模式，域模式	显示帮助信息并退出。
--host-config=<config>	域模式	要使用的主机配置文件的名称。默认是 host.xml 。
--interprocess-hc-address=<address>	域模式	主机控制器侦听的与进程控制器通讯的地址。
--interprocess-hc-port=<port>	域模式	主机控制器侦听的与进程控制器通讯的端口。
--master-address=<address>	域模式	设置系统属性 jboss.domain.master.address 为给定的值。在默认的从主机控制器配置里，它被用来配置和主主机控制器进行原生通讯的地址。
--master-port=<port>	域模式	设置系统属性 jboss.domain.master.port 为给定的值。在默认的从主机控制器配置里，它被用来配置和主主机控制器进行原生通讯的端口。
--read-only-server-config=<config>	独立服务器模式	要使用的服务器配置文件的名称。它和 --server-config 和 -c 不同，因为原始文件不会被覆盖。

参数或开关	模式	描述
--read-only-domain-config=<config>	域模式	要使用的域配置文件的名称。这和 --domain-config 和 -c 参数不同，因为初始文件不会被覆盖。
--read-only-host-config=<config>	域模式	要使用的主机配置文件的名称。这和 --host-config 参数不同，因为初始文件不会被覆盖。
-P <url> , -P=<url> , --properties=<url>	独立服务器模式，域模式	从给定的 URL 加载系统属性。
--pc-address=<address>	域模式	进程控制器与其控制的进程通讯时侦听的地址。
--pc-port=<port>	域模式	进程控制器与其控制的进程通讯时侦听的端口。
-S<name>[=<value>]	独立服务器模式	设置安全属性。
--server-config=<config>	独立服务器模式	要使用的服务器配置文件的名称。默认是 standalone.xml 。
-u <value> , -u=<value>	独立服务器模式，域模式	设置系统属性 jboss.default.multicast.address 为给定的值。
-v , -V , --version	独立服务器模式，域模式	显示应用服务器的版本并退出。

[提交 bug 报告](#)

2.2. 启动和停止服务器

2.2.1. 用管理 CLI 启动或停止服务器。

预备条件

- [第 3.5.2 节 “启动管理 CLI”](#)

您可以用管理 CLI 或管理控制台来启动和停止服务器（独立服务器模式）。在域模式里，您可以只启动服务器实例。这两个管理工具都允许您控制单个独立服务器实例，或者选择性地管理受管域部署里的多个服务器。如果您在域模式里使用管理控制台，请参考 [第 2.2.2 节 “使用管理控制台启动服务器”](#) 里的说明。如果您在使用管理 CLI，针对独立服务器和受管域实例的过程是不一样的。

用管理 CLI 启动或停止独立服务器。

独立服务器实例可以用命令行脚本启动，并用 **shutdown** 命令在管理 CLI 里关闭。如果你再次需要这个实例，请按照 [第 2.1.2 节 “将 JBoss EAP 6 作为独立服务器启动”](#) 里描述的过程再次启动。

例 2.6. 通过管理 CLI 停止独立服务器实例

```
[standalone@localhost:9999 /] shutdown
```

用管理 CLI 启动或停止受管域

如果你在运行受管域，管理控制台会允许你选择启动或关闭域里的特定服务器。这包括整个域里的服务器组，以及主机上的特定服务器实例。

例 2.7. 通过管理 CLI 停止受管域里的服务器主机

和独立服务器实例类似，**shutdown** 命令用于关闭声明的受管域主机。这个例子通过在调用关闭操作前声明实例名来关闭名为 **master** 的服务器主机。请用 **tab** 键来协助完成字符串并开放可见变量如可用的主机值。

```
[domain@localhost:9999 /] /host=master:shutdown
```

例 2.8. 通过管理 CLI 停止受管域里的服务器组

这个例子通过在调用 **start** 和 **stop** 操作前声明组启动了一个名为 **main-server-group** 的默认服务器组。请用 **tab** 键来协助完成字符串并开放可见变量如可用的主机组名的值。

```
[domain@localhost:9999 /] /server-group=main-server-group:start-servers
```

```
[domain@localhost:9999 /] /server-group=main-server-group:stop-servers
```

例 2.9. 通过管理 CLI 停止受管域里的服务器实例

这个例子通过在调用 **start** 和 **stop** 操作前声明主机启动和停止了 **master** 主机上的一个名为 **server-one** 的服务器实例。请用 **tab** 键来协助完成字符串并开放可见变量如可用的主机和服务器配置的值。

```
[domain@localhost:9999 /] /host=master/server-config=server-one:start
```

```
[domain@localhost:9999 /] /host=master/server-config=server-one:stop
```

[提交 bug 报告](#)

2.2.2. 使用管理控制台启动服务器

前提条件

- [第 2.1.3 节 “将 JBoss EAP 6 作为受管域启动”](#)
- [第 3.4.2 节 “登录到管理控制台”](#)

过程 2.10. 启动受管域里的服务器

1. 在控制器的顶部选择 **Runtime** 标签页。展开 **Server** 菜单并选择 **Overview**。

2. 从 **Server Instances** 列表里，选择要启动的服务器。正在运行的服务器会用一个复选框标记表示。

在这个列表的实例上悬停会在服务器细节下面以蓝色字体显示选项。

3. 要启动这个实例，请在 **Start Server** 文本出现时点击它。然后会出现一个确认对话框，点击 **Confirm** 按钮启动服务器。

结果

所选的服务器已启动并在运行中。

[提交 bug 报告](#)

2.2.3. 使用管理控制台停止服务器

前提条件

- [第 2.1.3 节 “将 JBoss EAP 6 作为受管域启动”](#)
- [第 3.4.2 节 “登录到管理控制台”](#)

过程 2.11. 使用管理控制台停止受管域里的服务器

1. 在控制器的顶部选择 **Runtime** 标签页。展开 **Domain** 菜单并选择 **Overview**。
2. 可用的 **Server Instances** 列表将显示在 **Hosts, groups and server instances** 表里。正在运行的服务器用一个复选框标记表示。
3. 将鼠标在所选的服务器上悬停。点击出现的 **Stop Server** 文本。然后将出现一个确认对话框。
4. 点击 **Confirm** 按钮来停止服务器。

结果

所选的服务器已停止。

[提交 bug 报告](#)

2.3. 文件系统路径

2.3.1. 文件系统路径

JBoss EAP 6 使用了文件系统路径的逻辑名称。**domain.xml**、**host.xml** 和 **standalone.xml** 配置都包含一个可以声明路径的部分。然后配置的其他部分可以通过逻辑名称引用这些路径，避免了为每个实例声明绝对路径。这有利于配置和管理，因为它允许将专有的主机配置解析为同一的逻辑名称。

例如，日志子系统配置包括对 **jboss.server.log.dir** 路径的引用，它指向服务器的 **log** 目录。

例 2.10. 日志目录的相对路路径示例

```
<file relative-to="jboss.server.log.dir" path="server.log"/>
```

JBoss EAP 6 自动提供大量的标准路径而无需用户在配置文件进行配置。

表 2.2. 标准路径

值	描述
<code>jboss.home.dir</code>	JBoss EAP 6 的根目录。
<code>user.home</code>	用户的主目录。
<code>user.dir</code>	用户的当前工作目录。
<code>java.home</code>	Java 的安装路径
<code>jboss.server.base.dir</code>	单独服务器实例的根目录。
<code>jboss.server.data.dir</code>	服务器用于持久性数据文件存储的目录。
<code>jboss.server.config.dir</code>	包含服务器配置的目录。
<code>jboss.server.log.dir</code>	服务器用于日志文件存储的目录。
<code>jboss.server.temp.dir</code>	服务器用于临时文件存储的目录。
<code>jboss.controller.temp.dir</code>	主机控制器用于临时文件存储的目录。

覆盖路径

如果您运行的是独立服务器，您可以以下面两种方式覆盖

`jboss.server.base.dir`、`jboss.server.log.dir` 或 `jboss.server.config.dir` 路径。

1. 您可以在启动服务器时传入命令行参数。例如：

```
bin/standalone.sh -Djboss.server.log.dir=/var/log
```

2. 您可以修改服务器配置文件里的 `JAVA_OPTS` 变量。请打开 `EAP_HOME/bin/standalone.conf` 文件并在结尾添加以下内容：

```
JAVA_OPTS="$JAVA_OPTS -Djboss.server.log.dir=/var/log"
```

运行在受管域里的服务器不支持路径覆盖。

添加自定义路径

您也可以创建自定义的路径。例如，您可以定义用于日志的相对路径：

■

```
my.relative.path=/var/log
```

然后您可以让日志处理程序使用 `my.relative.path`。

[提交 bug 报告](#)

2.4. 配置文件

2.4.1. 关于 JBoss EAP 6 配置文件

JBoss EAP 6 的配置和以前版本已经有了很大的修改。最显著的不同是使用了简化的配置文件结构，它包含下面列出的一个或多个文件。

表 2.3. 配置文件的位置

服务器模式	位置	目的
domain.xml	<i>EAP_HOME</i> /domain/configuration/domain.xml	这是受管域的主配置文件。只有域主控制器可以读取这个文件。对于其他成员，它可以被删除。
host.xml	<i>EAP_HOME</i> /domain/configuration/host.xml	这个文件包含了受管域里的物理主机专有的配置细节，如网络接口、套接字绑定、主机名称和其他主机专有的细节。 host.xml 文件包含了 host-master.xml 和 host-slave.xml 的全部功能，正如下面所描述的。这个文件没有出现在独立服务器里。
host-master.xml	<i>EAP_HOME</i> /domain/configuration/host-master.xml	这个文件包含了作为受管域里主服务器运行所需的配置细节。这个文件不会出现在独立服务器里。
host-slave.xml	<i>EAP_HOME</i> /domain/configuration/host-slave.xml	这个文件包含了作为受管域里从服务器运行所需的配置细节。这个文件不会出现在独立服务器里。
standalone.xml	<i>EAP_HOME</i> /standalone/configuration/standalone.xml	这是用于独立服务器的默认配置文件。它包含了独立服务器的所有信息，如子系统、网络、部署、套接字绑定和其他配置细节。当您启动独立服务器时会自动使用这个配置。

服务器模式	位置	目的
standalone-full.xml	<code>EAP_HOME/standalone/configuration/standalone-full.xml</code>	这是用于独立服务器的配置示例。它包含对每种可能的子系统的支持，除了那些要求高可用性的子系统。要使用它，请停止您的服务器并用下列命令重启： <code>EAP_HOME/bin/standalone.sh -c standalone-full.xml</code> 。
standalone-ha.xml	<code>EAP_HOME/standalone/configuration/standalone-ha.xml</code>	这个配置文件示例启用所有的默认子系统并为独立服务器添加了 <code>mod_cluster</code> 和 <code>JGroups</code> 子系统，所以它可以参与高可用性或负载均衡群集。这个文件不适用于受管域。要使用这个配置，请停止您的服务器并用下列命令重启： <code>EAP_HOME/bin/standalone.sh -c standalone-ha.xml</code> 。
standalone-full-ha.xml	<code>EAP_HOME/standalone/configuration/standalone-full-ha.xml</code>	这是用于独立服务器的配置示例。它包含对每种可能的子系统的支持，包含那些要求高可用性的子系统。要使用它，请停止您的服务器并用下列命令重启： <code>EAP_HOME/bin/standalone.sh -c standalone-full-ha.xml</code> 。

这些只是默认的位置。您可以在运行时指定不同的配置文件。

[提交 bug 报告](#)

2.4.2. 基于描述符的属性替换

应用程序配置 - 例如，数据源连接参数 - 通常在开发、测试和产品部署时都会不同。这有时是通过构建系统脚本实现的，因为 Java EE 规格没有包含表达这些配置的方法。

对于 JBoss EAP 6，您可以使用 *Descriptor-based property replacement* 在外部管理配置。

基于描述符的属性替换 (Descriptor-based property replacement) 基于描述符替换属性，允许您从应用程序和构建链里删除关于环境的假设。您可以在部署描述符而不是应用程序或构建系统脚本里指定环境专有的配置。您可以在文件里或作为命令行参数提供配置。

基于描述符的属性替换可以通过 **`standalone.xml`** 或 **`domain.xml`** 全局性地启用：

```
<subsystem xmlns="urn:jboss:domain:ee:1.1">
  <spec-descriptor-property-replacement>
    true
  </spec-descriptor-property-replacement>
  <jboss-descriptor-property-replacement>
```

```

    true
  </jboss-descriptor-property-replacement>
</subsystem>

```

ejb-jar.xml 和 **persistence.xml** 里的 Java EE 描述符可以被替换。这默认是禁用的。

JBoss 专有的描述符替换默认是启用的。描述符可以在以下文件里替换：

- **jboss-ejb3.xml**
- **jboss-app.xml**
- **jboss-web.xml**
- ***-jms.xml**
- ***-ds.xml**

例如，具有下列注解的 Bean：

```

@ActivationConfigProperty(propertyName = "connectionParameters",
propertyValue = "host=192.168.1.1;port=5445")

```

启用了基于描述符的属性替换后，**connectionParameters** 可以通过命令行来指定：

```

./standalone.sh -DconnectionParameters='host=10.10.64.1;port=5445'

```

通过系统属性实现相同功能：

```

<activation-config>
  <activation-config-property>
    <activation-config-property-name>
      connectionParameters
    </activation-config-property-name>
    <activation-config-property-value>
      ${jms.connection.parameters:'host=10.10.64.1;port=5445'}
    </activation-config-property-value>
  </activation-config-property>
</activation-config>

```

\${jms.connection.parameters:'host=10.10.64.1;port=5445'} 允许命令行提供的参数来覆盖连接参数，而且提供了默认值。

[提交 bug 报告](#)

2.4.3. 启用/禁用基于描述符的属性替换

介绍

jboss-as-ee_1_1.xsd 引入了对描述符属性替换的有限控制。本节内容涵盖了配置基于描述符的属性替换所需的步骤。

预备条件

- [第 2.1.1 节 “启动 JBoss EAP 6”](#)

- 第 3.5.2 节 “启动管理 CLI”

基于描述符的属性替换标记具有的布尔值：

- 如果设置为 **true**，属性替换将被启用。
- 如果设置为 **false**，属性替换将被禁用。

过程 2.12. jboss-descriptor-property-replacement

jboss-descriptor-property-replacement 用于在下列描述符里启用或禁用属性替换：

- **jboss-ejb3.xml**
- **jboss-app.xml**
- **jboss-web.xml**
- ***-jms.xml**
- ***-ds.xml**

jboss-descriptor-property-replacement 的默认值是 **true**。

1. 在管理 CLI 里，运行下列命令来确定 **jboss-descriptor-property-replacement** 的值：

```
/subsystem=ee:read-attribute(name="jboss-descriptor-property-replacement")
```

2. 运行下列命令来配置其行为：

```
/subsystem=ee:write-attribute(name="jboss-descriptor-property-replacement",value=VALUE)
```

过程 2.13. spec-descriptor-property-replacement

spec-descriptor-property-replacement 用于在下列描述符里启用或禁用属性替换：

- **ejb-jar.xml**
- **persistence.xml**

spec-descriptor-property-replacement 的默认值是 **false**。

1. 在管理 CLI 里，运行下列命令来确认 **spec-descriptor-property-replacement** 的值：

```
/subsystem=ee:read-attribute(name="spec-descriptor-property-replacement")
```

2. 运行下列命令来配置其行为：

```
/subsystem=ee:write-attribute(name="spec-descriptor-property-replacement",value=VALUE)
```

结果

成功地配置了基于描述符的属性替换标记。

[提交 bug 报告](#)

2.4.4. 配置文件历史

应用服务器的配置文件包括 **standalone.xml**、**domain.xml** 和 **host.xml**。虽然您可以直接编辑这些文件，我们推荐用可用的管理操作（如管理 CLI 或管理控制台）来配置应用服务器模型。

为了协助维护和管理服务器实例，应用服务器在启动时创建了原始配置文件的带时间戳的版本。管理操作导致的任何其他的配置修改都会让原始文件自动备份，而实例的一个工作备份会保留以供引用或回滚。这个归档功能可以扩展为保存、加载和删除服务器配置的快照，从而允许回调和回滚场景。

- [第 2.4.5 节 “用以前的配置启动服务器”](#)
- [第 2.4.6 节 “使用管理 CLI 保存配置快照”](#)
- [第 2.4.7 节 “使用管理 CLI 加载配置快照”](#)
- [第 2.4.8 节 “使用管理 CLI 删除配置快照”](#)
- [第 2.4.9 节 “使用管理 CLI 列出所有的配置快照”](#)

[提交 bug 报告](#)

2.4.5. 用以前的配置启动服务器

下面的例子展示了如何用独立服务器的 **standalone.xml** 里的以前的配置启动应用服务器。相同的概念也适用于受管域的 **domain.xml** 和 **host.xml**。

这个例子回调了管理操作修改服务器模型时应用服务器自动保存的配置。

1. 确定您要启动的备份版本。这个例子将回调成功引导后第一次修改前的服务器模型的实例。

```
EAP_HOME/standalone/configuration/standalone_xml_history/current/standalone.v1.xml
```

2. 传入 **jboss.server.config.dir** 下的相对文件名，用备份模型的配置启动服务器。

```
EAP_HOME/bin/standalone.sh --server-config=standalone_xml_history/current/standalone.v1.xml
```

结果

应用服务器用所选的配置启动了。



注意

域配置历史位于

EAP_HOME/domain/configuration/domain_xml_history/current/domain.v1.xml

传入 **jboss.domain.config.dir** 下的相对文件名，用备份模型的配置启动服务器。

用下面的配置启动域：

```
EAP_HOME/bin/domain.sh --domain-
config=domain_xml_history/current/domain.v1.xml
```

[提交 bug 报告](#)

2.4.6. 使用管理 CLI 保存配置快照

概述

配置快照是当前服务器配置的时间点拷贝。管理员可以保存和加载这些拷贝。

下面的例子使用了 **standalone.xml** 配置文件，但相同的过程适用于 **domain.xml** 和 **host.xml** 配置文件。

必须具备的条件

- [第 3.5.2 节“启动管理 CLI”](#)

过程 2.14. 创建配置快照并保存

- **保存快照**
运行 **take-snapshot** 操作来创建当前服务器配置的快照。

```
[standalone@localhost:9999 /] :take-snapshot
{
  "outcome" => "success",
  "result" =>
"/home/User/EAP_HOME/standalone/configuration/standalone_xml_history
/snapshot/20110630-172258657standalone.xml"
```

结果

保存了当前服务器配置的快照。

[提交 bug 报告](#)

2.4.7. 使用管理 CLI 加载配置快照

配置快照是当前服务器配置的时间点拷贝。管理员可以保存和加载这些拷贝。加载快照的过程和用于 [第 2.4.5 节“用以前的配置启动服务器”](#) 的方法类似，都是从命令行而不是管理 CLI 界面运行来创建、列出和删除快照。

下面的例子使用了 **standalone.xml** 配置文件，但相同的过程适用于 **domain.xml** 和 **host.xml** 配置文件。

过程 2.15. 加载配置快照

1. 确定要加载的快照。这个例子将从 **snapshot** 目录回调下列文件。下面是快照文件的默认路径。

```
EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20110812-191301472standalone.xml
```

快照是用相对路径表达的，上面的例子可以像下面这样编写。

```
jboss.server.config.dir/standalone_xml_history/snapshot/20110812-191301472standalone.xml
```

2. 通过传入文件名用所选的配置快照启动服务器。

```
EAP_HOME/bin/standalone.sh --server-config=standalone_xml_history/snapshot/20110913-164449522standalone.xml
```

结果

服务器用加载快照里选择的配置进行了重启。

[提交 bug 报告](#)

2.4.8. 使用管理 CLI 删除配置快照

必须具备的条件

- [第 3.5.2 节 “启动管理 CLI”](#)

配置快照是当前服务器配置的时间点拷贝。管理员可以保存和加载这些拷贝。

下面的例子使用了 **standalone.xml** 配置文件，但相同的过程适用于 **domain.xml** 和 **host.xml** 配置文件。

过程 2.16. 删除专有的快照

1. 确定要删除的快照。这个例子将从 **snapshot** 目录删除下列文件。

```
EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20110630-165714239standalone.xml
```

2. 如下例所示，指定快照的名称，运行 **:delete-snapshot** 命令来删除专有的快照。

```
[standalone@localhost:9999 /] :delete-snapshot(name="20110630-165714239standalone.xml")
{"outcome" => "success"}
```

结果

快照已被删除。

过程 2.17. 删除所有快照

- 如下例所示，运行 `:delete-snapshot(name="all")` 命令删除所有的快照。

```
[standalone@localhost:9999 /] :delete-snapshot(name="all")
{"outcome" => "success"}
```

结果

所有快照都已被删除。

[提交 bug 报告](#)

2.4.9. 使用管理 CLI 列出所有的配置快照

必须具备的条件

- [第 3.5.2 节“启动管理 CLI”](#)

配置快照是当前服务器配置的时间点拷贝。管理员可以保存和加载这些拷贝。

下面的例子使用了 `standalone.xml` 配置文件，但相同的过程适用于 `domain.xml` 和 `host.xml` 配置文件。

过程 2.18. 列出所有的配置快照

- **列出所有快照**

用 `:list-snapshots` 命令列出所有保存的快照。

```
[standalone@localhost:9999 /] :list-snapshots
{
  "outcome" => "success",
  "result" => {
    "directory" =>
"/home/hostname/EAP_HOME/standalone/configuration/standalone_xml_his
tory/snapshot",
    "names" => [
      "20110818-133719699standalone.xml",
      "20110809-141225039standalone.xml",
      "20110802-152010683standalone.xml",
      "20110808-161118457standalone.xml",
      "20110912-151949212standalone.xml",
      "20110804-162951670standalone.xml"
    ]
  }
}
```

结果

快照被列出。

[提交 bug 报告](#)

第 3 章 管理接口

3.1. 管理应用服务器

JBoss EAP 6 提供了多个管理工具来配置和管理你的实现。其中包括新的管理控制台和管理命令行界面 (CLI)，以及让专家用户可以开发自己的工具的底层管理 API。

[提交 bug 报告](#)

3.2. 管理应用程序编程接口 (API)

管理客户

JBoss EAP 6 提供了不同的方法来配置和管理服务器，您可以使用 web 界面、命令行客户或一系列的 XML 配置文件。我们推荐的方法是编辑配置文件，包括管理控制台和管理 CLI。对配置文件的编辑总会在不同视图里同步并最终持久化到 XML 文件里。请注意，在服务器实例运行时对 XML 配置文件的修改将被服务器模型所覆盖。

HTTP API

管理控制台是用 Google Web Toolkit (GWT) 构建的 Web 界面的例子。管理控制台和服务器通过 HTTP 管理接口进行通讯。HTTP API 端点是依赖于 HTTP 协议来集成管理层的管理客户的入口点。它使用 JSON 编码协议和 de-typed RPC 风格的 API 来描述和执行管理操作。HTTP API 用于 Web 控制台，它也为许多其他客户提供了集成能力。

HTTP API 端点和主机控制器或独立服务器是共存的。HTTP API 端点服务两种不同的上下文，一个用于执行管理操作，另一个用于访问 Web 界面。在默认情况下，它运行在端口 9990 上。

例 3.1. HTTP API 配置文件示例

```
<management-interfaces>
  [...]
  <http-interface security-realm="ManagementRealm">
    <socket-binding http="management-http"/>
  </http-interface>
</management-interfaces>
```

Web 控制台是通过和 HTTP 管理 API 相同的端口来访问的。区别作为默认的本地主机访问的管理控制台、通过专门主机和端口远程访问的管理控制台和开放的域 API 是很重要的。

表 3.1. 访问管理控制台的 URL

URL	描述
<code>http://localhost:9990/console</code>	访问本地主机的管理控制台，它控制受管域的配置。
<code>http://hostname:9990/console</code>	远程访问管理控制台的主机命名和受管域配置。
<code>http://hostname:9990/management</code>	和管理控制台运行在相同端口上的 HTTP 管理 API，它显示原始属性和开放给 API 的值。

Native API

管理 CLI 就是 Native API 工具的一个例子。这个管理工具可用于受管域及独立服务器，它允许用户连接到域控制器或独立服务器实例并执行 de-typed 管理模型里可用的管理操作。

Native API 端点是依赖于原生协议来集成管理层的管理客户的入口点。它使用开放的二进制协议和基于非常少量的 Java 类型的 RPC 风格的 API 来描述和执行管理操作。它被管理 CLI 工具所使用，但也为很多其他客户提供了集成能力。

Native API 端点和主机控制器或独立服务器是共存的。使用管理 CLI 必须启用它。在默认情况下，它运行在端口 9999 上。

例 3.2. Native API 配置文件示例

```
<management-interfaces>
  <native-interface security-realm="ManagementRealm">
    <socket-binding native="management-native"/>
  </native-interface>
  [...]
</management-interfaces>
```

[提交 bug 报告](#)

3.3. 关于管理控制台和管理 CLI

在 JBoss EAP 6 里，所有的服务器实例和配置都通过管理界面而不是编辑 XML 文件来进行配置。虽然 XML 配置文件仍可进行编辑，通过管理界面的管理提供了额外校验和服务器实例持久性管理的高级功能。服务器运行时对 XML 配置文件的修改仍会被服务器模型覆盖，任何添加的 XML 注释都会被删除。当服务器实例在运行时，我们应该只使用管理界面来修改配置文件。

要通过 Web 浏览器里的图形化用户界面来管理服务器，请使用管理控制台。

要通过命令行界面来管理服务器，则请使用管理 CLI。

[提交 bug 报告](#)

3.4. 管理控制台

3.4.1. 管理控制台

管理控制台是用于 JBoss EAP 6 的基于 Web 的管理工具。

您可以使用管理控制台来启动和停止服务器、部署和卸载应用程序、调整系统设置并对服务器配置进行持久性修改。管理控制台也可以执行管理任务，并在修改需要服务器实例重启或重载时进行实时通知。

在受管域里，相同域里的服务器实例和服务器组可以通过域控制器的管理控制台进行集中管理。

[提交 bug 报告](#)

3.4.2. 登录到管理控制台

Red Hat JBoss EAP 管理控制台的主页为独立服务器和运行在受管域里的服务器提供了一个统一的入口点。这个页面也提供到重要的开发人员、操作性资源和普通资源的链接。

预备条件

- 您必须创建一个管理型用户，请参考：[第 4.1.1 节 “为管理接口添加用户”](#)。

JBoss EAP 6 服务器必须正在运行。

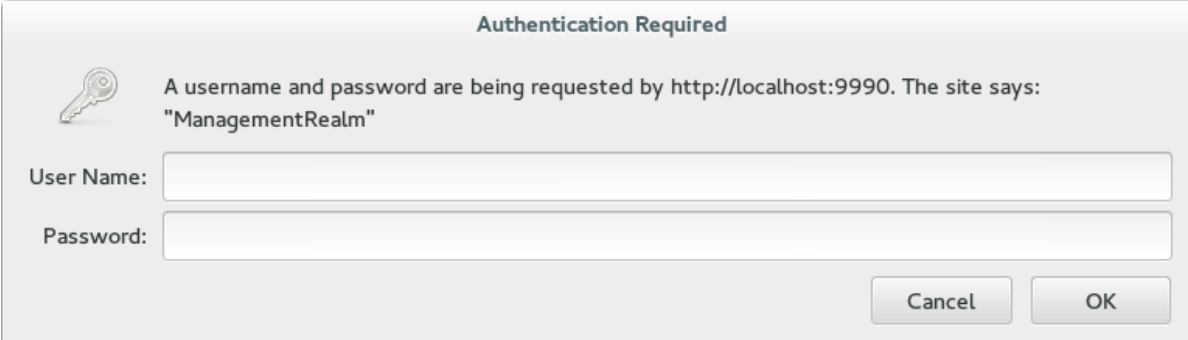
过程 3.1. 登录到管理控制台

1. 进入管理控制台开始页面

通过 Web 浏览器进入管理控制台。默认的位置是 <http://localhost:9990/console/App.html>，其中的 9990 是预定义为管理控制台的套接绑定的。

2. 登录到管理控制台

输入您之前创建的帐号的用户名和密码来登录到管理控制台屏幕。

A screenshot of a web browser's authentication dialog box. The title bar says "Authentication Required". Inside, there is a key icon and text: "A username and password are being requested by http://localhost:9990. The site says: 'ManagementRealm'". Below this, there are two input fields: "User Name:" and "Password:". At the bottom right, there are two buttons: "Cancel" and "OK".

Authentication Required

A username and password are being requested by http://localhost:9990. The site says: "ManagementRealm"

User Name:

Password:

Cancel OK

图 3.1. 管理控制台的登录屏幕

结果

登录后，下面的管理控制台登录页面将出现：<http://localhost:9990/console/App.html#home>

[提交 bug 报告](#)

3.4.3. 修改管理控制台的语言

基于 Web 的管理控制台默认使用英语，但您可以选择下列语言。

所支持的语言

- 德语 (de)
- 简体中文 (zh-Hans)
- 巴西葡萄牙语 (pt-BR)
- 法语 (fr)
- 西班牙语 (es)
- 日语 (ja)

过程 3.2. 修改基于 Web 的管理控制台的语言

1. 登录到管理控制台。

登录到基于 Web 的管理控制台。

2. 打开 Setting 对话框。

屏幕的右下角是 **Settings** 标签。点击它打开管理控制台的设置对话框。

3. 选择语言。

在 **Locale** 选择框里选择语言。然后点击 **Save**。确认框通知您需要重载应用程序。点击 **Confirm**。刷新 Web 浏览器以使用新的区域设置。

[提交 bug 报告](#)

3.4.4. EAP 控制台里的数据分析

关于 Google Analytics

Google Analytics 是一个提供网站的完整使用统计的免费 Web 分析服务。它提供网站的访问者的关键数据，如访问次数、页面查看、每次访问的页面数及平均逗留时间等。Google Analytics 也提供网站的知名度和使用情况。

关于 EAP 管理控制台里的 Google Analytics

JBoss EAP 6.3 为用户提供了启用/禁用管理控制台里 Google Analytics 的选项。Google Analytics 可帮助 Red Hat EAP 团队理解用户如何使用控制台以及哪些部分是最受用户关注的。这些信息反过来可帮助 EAP 团队调整控制台的设计、功能、内容以适应用户的需求。

[提交 bug 报告](#)

3.4.5. 启用/禁用 EAP 控制台里的 Google Analytics

要启用 EAP 管理控制台里的 Google Analytics：

- 登录到管理控制台
- 点击控制台右下角的 **Settings** 按钮。

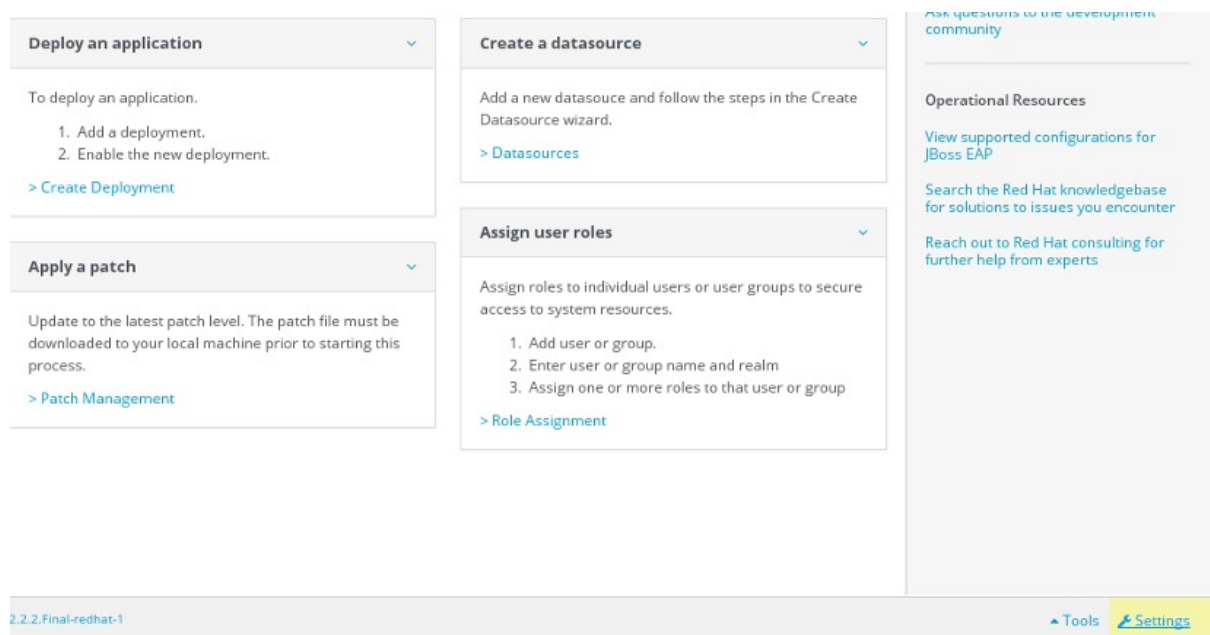
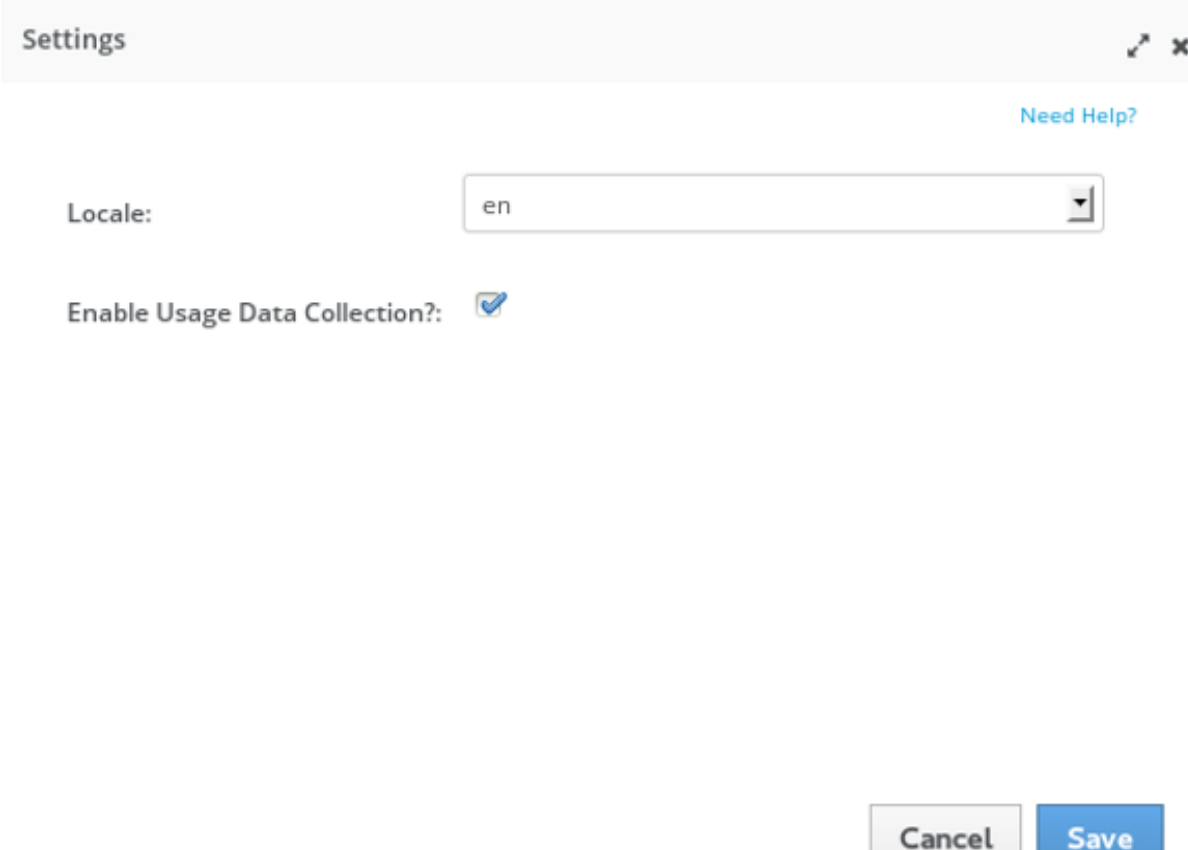


图 3.2. 管理控制台的登录屏幕

- 选择 **Settings** 窗口上的 **Enable Usage Data Collection** 复选框并点击 **Save** 按钮。重载应用程序以激活新的设置。



The screenshot shows a 'Settings' dialog box with a title bar containing the word 'Settings' and window control icons (maximize, close). In the top right corner of the dialog, there is a blue link that says 'Need Help?'. The main content area has two settings: 'Locale:' followed by a dropdown menu currently showing 'en', and 'Enable Usage Data Collection?:' followed by a checked checkbox. At the bottom right, there are two buttons: a grey 'Cancel' button and a blue 'Save' button.

图 3.3. 设置窗口（启用使用数据收集）

要在启用后再禁用管理控制台的 Google Analytics，请点击 **Settings** 窗口上的 **Disable Usage Data Collection** 选项，然后点击 **Save**。

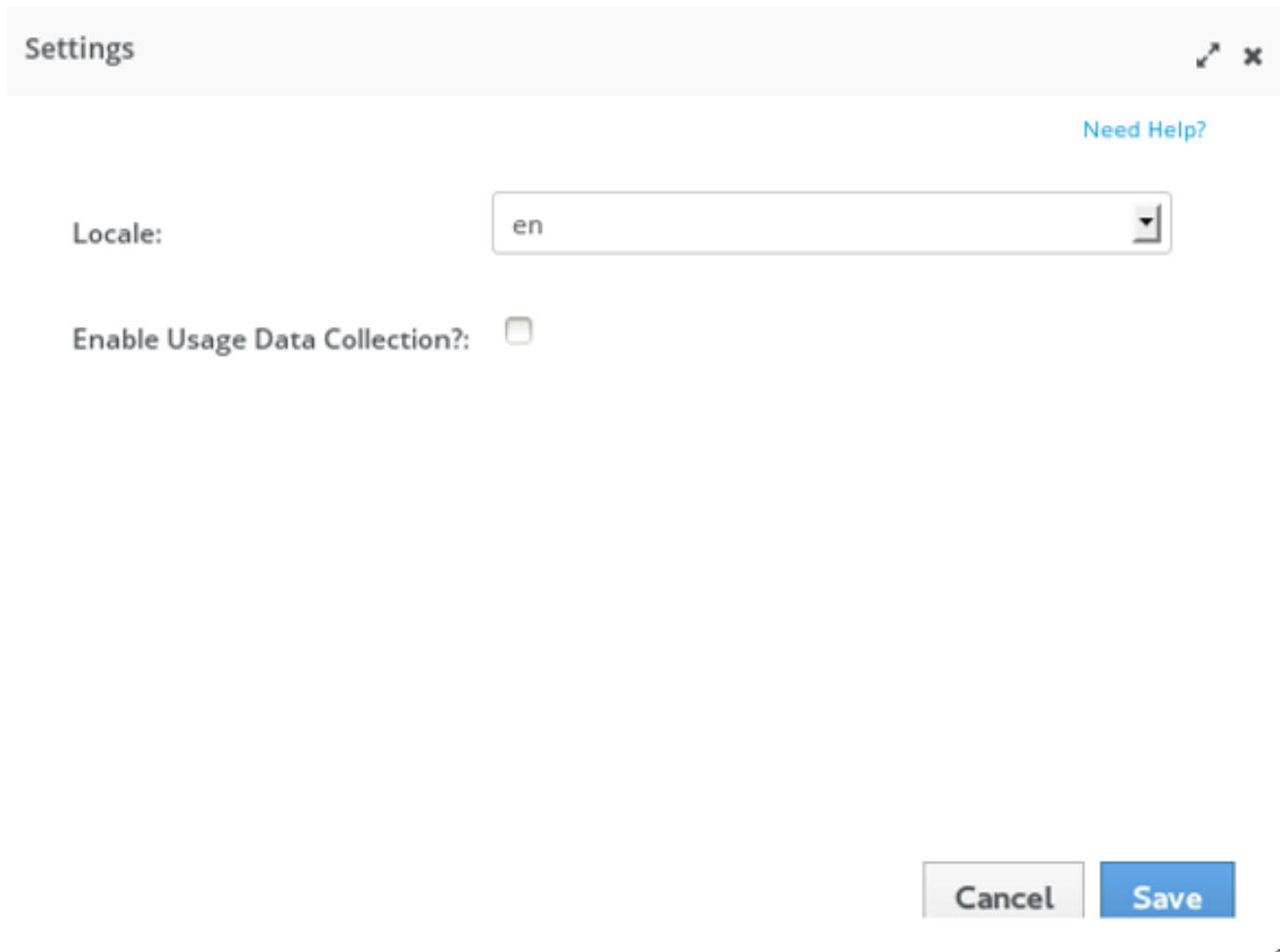


图 3.4. 设置窗口（禁用使用数据收集）



注意

EAP 6.3 默认是禁用 Google Analytics 的，使用它是可选的。

[提交 bug 报告](#)

3.4.6. 使用管理控制台配置服务器

前提条件

- [第 2.1.3 节 “将 JBoss EAP 6 作为受管域启动”](#)
- [第 3.4.2 节 “登录到管理控制台”](#)

过程 3.3. 配置服务器

1. 从控制台的顶部选择 **Domain** 标签页。可用的服务器将显示在表格里。
2. 从 **Available Server Configurations** 表格里选择服务器实例。
3. 在所选的服务器细节上方点击 **Edit** 按钮。
4. 修改配置属性。

5. 点击**Save**完成。

结果

服务器配置被修改，且会在服务器重启时生效。

[提交 bug 报告](#)

3.4.7. 在管理控制台里添加部署

前提条件

- [第 3.4.2 节 “登录到管理控制台”](#)
1. 从控制台的顶部选择 **Runtime** 标签页。
 2. 对于独立服务器，展开 **Server** 菜单项并选择 **Manage Deployments**。对于受管域，展开 **Domain** 菜单项并选择 **Manage Deployments**。**Manage Deployments** 面板将显示。
 3. 点击 **Content Repository** 标签页上的 **Add** 按钮。**Create Deployment** 对话框将会出现。
 4. 在这个对话框里，点击 **Browse** 按钮。选择您要部署和上传的文件。然后点击 **Next** 按钮进行。
 5. 验证出现在 **Create Deployments** 对话框里的部署名和 runtime 名称。验证名称后请点击 **Save** 按钮上传文件。

结果

所选的内容被上传至服务器且可以进行部署了。

[提交 bug 报告](#)

3.4.8. 在管理控制台里创建新的服务器

前提条件

- [第 2.1.3 节 “将 JBoss EAP 6 作为受管域启动”](#)
- [第 3.4.2 节 “登录到管理控制台”](#)

过程 3.4. 创建新的服务器配置

1. **进入管理控制台里的 Server Configurations 页面**
从控制台的顶部选择 **Domain** 标签页。
2. **创建新的配置**
 - a. 点击 **Available Server Configuration** 表上方的 **Add** 按钮。
 - b. 在 **Create Server Configuration** 对话框里输入基本服务器设置。
 - c. 点击 **保存** 按钮保存新的服务器配置。

结果

新服务器被创建且出现在 **Server Configurations** 列表里。

[提交 bug 报告](#)

3.4.9. 用管理控制台修改默认的日志级别

过程 3.5. 编辑日志级别

1. 进入管理控制台的 Logging 面板

- a. 对于受管域，选择控制台顶部的 **Configuration** 标签页，然后从控制台左侧的下拉菜单里选择相关的配置集。
- b. 对于受管域或独立服务器，展开控制台左侧菜单的 **Core** 菜单并点击 **Logging** 条目。
- c. 点击控制台顶部的 **Log Categories** 标签页。

2. 编辑 logger 细节

编辑 **Log Categories** 表格里条目的细节。

- a. 在 **Log Categories** 表格里选择条目，然后点击下面的 **Details** 部分里的 **Edit** 按钮。
- b. 通过 **Log Level** 下拉框选择类别的日志级别。完成后点击 **Save** 按钮。

结果

相关类别的日志级别已被更新。

[提交 bug 报告](#)

3.4.10. 在管理控制台里创建新的服务器组

必须具备的条件

- [第 3.4.2 节 “登录到管理控制台”](#)

过程 3.6. 配置和添加新的服务器组

1. 进入 Server Groups 视图

从控制台的顶部选择 **Domain** 标签页。

2. 展开左侧菜单里的 Server 标签。选择 Server Groups。

3. 添加服务器组

点击 **Add** 按钮来添加新的服务器组。

4. 配置服务器组

- a. 输入服务器组的名称。
- b. 选择服务器组的配置集。
- c. 选择服务器组的套接字绑定。
- d. 点击 **Save** 按钮来保存新的组。

结果

新创建的服务器组出现在管理控制台里了。

[提交 bug 报告](#)

3.5. 管理 CLI

3.5.1. 关于管理命令行接口（**Command Line Interface**，**CLI**）

管理 CLI 是 JBoss EAP 6 的一个命令行管理工具。

使用管理 CLI 启动和停止服务器、部署和卸载应用程序、配置系统设置并执行其他管理任务。操作可以批量模式进行，将多个任务作为一个组来运行。

[提交 bug 报告](#)

3.5.2. 启动管理 CLI

前提条件：

- [第 2.1.2 节 “将 JBoss EAP 6 作为独立服务器启动”](#)
- [第 2.1.3 节 “将 JBoss EAP 6 作为受管域启动”](#)

过程 3.7. 在 Linux 或 Windows 里启动 CLI

- 在 Linux 里启动 CLI
运行 ***EAP_HOME/bin/jboss-cli.sh*** 命令：
- 在 Windows 里启动 CLI
运行 ***EAP_HOME\bin\jboss-cli.bat*** 命令：

```
$ EAP_HOME/bin/jboss-cli.sh
```

```
C:\>EAP_HOME\bin\jboss-cli.bat
```

[提交 bug 报告](#)

3.5.3. 退出管理 CLI

前提条件

- [第 3.5.2 节 “启动管理 CLI”](#)

过程 3.8. 退出管理 CLI

- 运行 **quit** 命令
在管理 CLI 里，输入 **quit** 命令：

```
[domain@localhost:9999 /] quit
```

[提交 bug 报告](#)

3.5.4. 用管理 CLI 连接受管服务器实例

前提条件

- [第 3.5.2 节 “启动管理 CLI”](#)

过程 3.9. 连接至受管服务器实例

- **运行 connect 命令**

在管理 CLI 里，输入 **connect** 命令：

```
[disconnected /] connect
Connected to domain controller at localhost:9999
```

- 或者，在 Linux 系统里启动管理 CLI 时使用 **--connect** 参数来连接至受管服务器：

```
$ EAP_HOME/bin/jboss-cli.sh --connect
```

- **--connect** 参数可以用来指定服务器的主机和端口。要连接至地址 **192.168.0.1** 和端口 **9999**，请使用下列命令：

```
$ EAP_HOME/bin/jboss-cli.sh --connect --
controller=192.168.0.1:9999
```

[提交 bug 报告](#)

3.5.5. 用管理 CLI 获取帮助

概述

管理 CLI 有带有普通和上下文敏感选项的帮助对话框。对于独立服务器和域控制器，依赖于操作上下文的 **help** 命令都要求已建立连接。除非连接已建立，否则这些命令不会出现在列表里。

必须具备的条件

- [第 3.5.2 节 “启动管理 CLI”](#)

过程 3.10. 普通和上下文敏感帮助

1. **运行 help 命令**

在管理 CLI 里，输入 **help** 命令：

```
[standalone@localhost:9999 /] help
```

2. **获取上下文敏感帮助**

在管理 CLI 里，输入 **help -commands** 扩展命令：

```
[standalone@localhost:9999 /] help --commands
```

3. 关于特定命令的更多详情，请运行 **help** 并以 **'--help'** 为参数。

```
[standalone@localhost:9999 /] deploy --help
```

■

结果

CLI 帮助信息被显示。

[提交 bug 报告](#)

3.5.6. 以批模式使用管理 CLI

介绍

批处理允许大量的操作请求按序列进行分组，然后作为一个单元来执行。如果序列里的任何以一个操作请求执行失败，整个操作组都将回滚。

前提条件

- [第 3.5.2 节 “启动管理 CLI”](#)
- [第 3.5.4 节 “用管理 CLI 连接受管服务器实例”](#)

过程 3.11. 批处理模式命令和操作

1. 启用批处理模式

用 **batch** 命令启用批处理模式。

```
[standalone@localhost:9999 /] batch
[standalone@localhost:9999 / #]
```

提示里的井号（#）指明了批处理模式

2. 添加操作请求到批处理命令里

在批处理模式下，照常输入操作请求。操作请求将按输入的顺序添加到批处理命令里。

关于格式化操作请求的详情，请参考 [第 3.5.8 节 “在管理 CLI 里使用操作和命令”](#)。

3. 运行批处理命令

一旦输入了整个操作请求序列，请用 **run-batch** 运行这个批处理命令。

```
[standalone@localhost:9999 / #] run-batch
The batch executed successfully.
```

关于可用于批处理的完整的命令列表，请参考 [第 3.5.7 节 “CLI 批处理模式命令”](#)。

4. 保存在外部文件里的批命令

频繁运行的批处理命令可以存储在外部文件里，通过将完整文件路径作为参数传入 **batch** 或直接作为 **run-batch** 命令的参数来执行。

您可以用文本编辑器创建批处理命令文件。每个命令都必须单独一行且 CLI 应该可以访问它。

下面的命令将加载 **myscript.txt** 文件至批处理模式。这个文件里的所有命令都可以编辑或删除，您也可以插入新的命令。这个批处理会话里进行的修改不会持久化到 **myscript.txt** 文件里。

```
[standalone@localhost:9999 /] batch --file=myscript.txt
```


下面的命令将立即运行存储在文件 `myscript.txt` 里的批命令

```
[standalone@localhost:9999 /] run-batch --file=myscript.txt
```

结果

输入的操作请求序列以批模式完成了。

[提交 bug 报告](#)

3.5.7. CLI 批处理模式命令

这个表提供了 JBoss EAP 6 CLI 里可用的批处理命令列表。这些命令只能用于批处理模式。

表 3.2. CLI 批处理模式命令

命令名	描述
list-batch	当前批次里的命令和操作的列表。
edit-batch-line line-number edited-command	通过行号和要编辑的命令来编辑当前批处理里的行。 例如： edit-batch-line 2 data-source disable --name=ExampleDS 。
move-batch-line fromline toline	指定您要移动的行号为第一个参数且新的位置为第二个参数来重新对批处理命令里的行进行排序。例如： move-batch-line 3 1 。
remove-batch-line linenummer	删除指定行上的批处理命令。例如： remove-batch-line 3 。
holdback-batch [batchname]	<p>如果您突然想在 CLI 里执行批处理之外的命令，您可以使用这个命令推迟或存储当前的批处理命令。要返回已暂停批模式，只要在 CLI 命令行上再次输入 batch 就可以了。</p> <p>如果在 holdback-batch 命令时您提供了 batchname，批命令将按照这个名称进行存储要返回命名的批次，请使用 batch batchname。不使用 batchname 调用 batch 命令将启动新的（未命名）的批处理。系统里只能有一个未命名的暂停批命令。</p> <p>要查看暂停批命令的列表，请使用 batch -l 命令。</p>
discard-batch	取消当前活动的批处理命令。

[提交 bug 报告](#)

3.5.8. 在管理 CLI 里使用操作和命令

前提条件

- 第 3.5.2 节 “启动管理 CLI”

- 第 3.5.4 节 “用管理 CLI 连接受管服务器实例”

过程 3.12. 创建、配置和执行请求

1. 构造操作请求

操作请求允许和管理模型的低层交互。它们提供一种可控的方式来编辑服务器配置。操作请求由三部分组成：

- *地址*，前缀为斜杠 (/)。
- *操作名称*，前缀为冒号 (:)。
- 可选的参数，包含在括号 (()) 里。

a. 确定地址

配置以有地址的资源层级树型出现。每个资源节点都提供了一系列不同的操作。地址指定哪些资源可以执行操作。地址使用下面的语法：

```
/node-type=node-name
```

- *node-type* 是资源节点的类型。它映射配置 XML 文件里的元素名称。
- *node-name* 是资源节点的名称。它映射配置 XML 文件里的元素的 **name** 属性。
- 用斜杠 (/) 分隔资源树的每个级别。

要确定所需的地址，请参考 XML 配置文件。**EAP_HOME/standalone/configuration/standalone.xml** 文件保存独立服务器的配置信息，**EAP_HOME/domain/configuration/domain.xml** 和 **EAP_HOME/domain/configuration/host.xml** 文件保存受管域的配置信息。

例 3.3. 操作地址示例

要执行 Logging 子系统上的操作，请使用操作请求里的下列地址：

```
/subsystem=logging
```

要执行 Java 数据源上的操作，请使用操作请求里的下列地址：

```
/subsystem=datasources/data-source=java
```

b. 确定操作

对于不同类型的资源节点，操作会有所不同。操作使用下面的语法：

```
:operation-name
```

- *operation-name* 是要请求的操作的名称。

在独立服务器上的任何资源地址上使用 **read-operation-names** 操作来列出可用的操作。

例 3.4. 可用的操作

要列出 Logging 子系统到所有可用的操作，在独立服务器里输入下列请求：

```
[standalone@localhost:9999 /] /subsystem=logging:read-
operation-names
{
  "outcome" => "success",
  "result" => [
    "add",
    "read-attribute",
    "read-children-names",
    "read-children-resources",
    "read-children-types",
    "read-operation-description",
    "read-operation-names",
    "read-resource",
    "read-resource-description",
    "remove",
    "undefine-attribute",
    "whoami",
    "write-attribute"
  ]
}
```

c. 确定任何参数

每个操作可能需要不同的参数。

参数使用下面的语法：

```
(parameter-name=parameter-value)
```

- *parameter-name* 是参数的名称。
- *parameter-value* 是参数的值。
- 多个参数用逗号隔开 (,)。

要确定所需的参数，在资源节点上执行 **read-operation-description** 命令，将操作名称作为参数传入。详情请参考 [例 3.5 “确定操作的参数”](#)。

例 3.5. 确定操作的参数

要确定 logging 子系统上的 **read-children-types** 操作的必需参数，请输入 **read-operation-description** 命令：

```
[standalone@localhost:9999 /] /subsystem=logging:read-
operation-description(name=read-children-types)
{
  "outcome" => "success",
  "result" => {
    "operation-name" => "read-children-types",
    "description" => "Gets the type names of all the
children under the selected resource",
    "reply-properties" => {
      "type" => LIST,
```

```

        "description" => "The children types",
        "value-type" => STRING
    },
    "read-only" => true
}
}

```

2. 输入完整的操作请求

一旦确定了地址、操作和所有参数，请输入完整的操作请求。

例 3.6. 操作请求示例

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-
resource(recursive=true)
```

结果

管理接口执行服务器配置里的操作请求。

[提交 bug 报告](#)

3.5.9. 管理 CLI 配置选项

每次 CLI 启动时都会加载其配置文件 - `jboss-cli.xml`。它必须位于 `$EAP_HOME/bin` 或者系统属性 `jboss.cli.config` 指定的目录。

default-controller

如果不带参数执行 `connect` 命令时连接的配置。

default-controller Parameters

host

控制器的主机名。默认值：`localhost`。

port

连接控制器的端口号码。默认值为 `9999`。

validate-operation-requests

指定在操作请求发往控制器执行前是否检验操作请求的参数列表。类型：`Boolean`；默认值：`true`。

history

这个元素包含命令和操作历史日志的配置。

history 参数

enabled

指定是否启用 `history`。类型：`Boolean`；默认值：`true`。

file-name

存储历史日志的文件的名称。默认值是：**.jboss-cli-history**。

file-dir

存储历史日志的目录。默认值是：**\$USER_HOME**。

max-size

历史日志文件的最大大小。默认值为 **500**。

resolve-parameter-values

是否在发送操作请求到控制器前解析指定为命令参数（或操作参数）的系统属性，或者让解析在服务器端进行。类型：**Boolean**；默认值：**false**。

connection-timeout

用控制器建立连接所允许的时间。类型：**Integer**；默认值：**5,000** 秒。

ssl

这个元素包含用于 SSL 的密钥和信任库的配置。

ssl 参数**vault**

类型：**vaultType**

key-store

类型：**String**

key-store-password

类型：**String**

alias

类型：**String**

key-password

类型：**String**

trust-store

类型：**String**

trust-store-password

类型：**String**

modify-trust-store

如果设置为 **false**，当接收到不承认的证书时 CLI 将提示用户并允许将它们存储在信任库。类型：**Boolean**；默认值：**true**。

vaultType

如果既没有指定 **code** 也没指定 **module**，默认的实现将被使用。如果指定了 **code** 但没有指定 **module**，系统将在 **Picketbox** 模块里查找指定的类。如果 **module** 和 **code** 都指定了，系统将在 'module' 指定的模块里查找 'code' 指定的类。

code

类型：String

module

类型：String

silent

指定信息和错误消息是否输出到终端。如果配置允许或者在命令行用 **>** 指定了输出目标，即使指定了 **false**，消息仍会用 **logger** 进行记录。

[提交 bug 报告](#)

3.5.10. 管理 CLI 命令参考

必须具备的条件

- [第 3.5.2 节 “启动管理 CLI”](#)

概述

[第 3.5.5 节 “用管理 CLI 获取帮助”](#) 描述了如何访问管理 CLI 的帮助功能，包括有带有普通和上下文敏感选项的帮助对话框。对于独立服务器和域控制器，依赖于操作上下文的 **help** 命令都要求已建立连接。除非连接已建立，否则这些命令不会出现在列表里。

表 3.3.

命令	描述
batch	通过创建新的批次、或者重新激活现有的暂停的批次来启动批模式。如果没有暂停的批命令，这个命令将启动新的批次。如果存在未命名的暂停的批命令，这个命令将重新激活它。如果存在有名称的暂停的批命令，将这个批次的名称作为参数来执行命令就可以激活。
cd	根据参数修改当前的节点路径。当前的节点路径用于不包含地址部分的操作请求的地址。如某个操作请求包含了地址，所包含的地址将当作当前节点路径的相对地址。当前的节点路径可以以节点类型结尾。此时，执行指定节点名称的操作就足够了，例如 logging:read-resource 。
clear	清除屏幕。
command	允许您添加、删除和列出现有的普通类型的命令。普通类型命令是分配专有节点类型的命令，它允许您执行该类型的实例的任何可用的操作。它也可以修改现有实例上类型开放的任何属性。

命令	描述
connect	连接到指定主机和端口上的控制器。
connection-factory	定义连接工厂。
data-source	管理 datasource 子系统里的 JDBC 数据源配置。
deploy	部署用文件路径指定的应用程序或启用资料库里现有的被禁用的应用程序。如果不带参数执行，这个命令将列出全部现有的部署。
help	显示帮助信息。它可以用 --commands 参数为给定命令提供上下文敏感的内容。
history	显示内存里的 CLI 命令历史以及启用/禁用历史扩展的状态。它可以按需要参数来清除、禁用和启用历史扩展。
jms-queue	在 messaging 子系统里定义一个 JMS 队列。
jms-topic	在 messaging 子系统里定义一个 JMS 主题。
ls	列出节点路径的内容。在默认情况下，终端窗口会用整屏以列显示结果。 -l 参数将以每行一个名字显示结果。
pwd	显示当前工作节点的完整节点路径。
quit	终止命令行界面。
read-attribute	根据参数显示受管资源属性的值和描述。
read-operation	显示指定操作的描述，未指定则列出所有的操作。
undeploy	以应用程序的名称为参数运行可以卸载应用程序。它也可以通过参数运行从资料库删除应用程序。如无参数运行则输出所有现有的部署。
version	显示应用服务器版本和环境信息。
xa-data-source	管理 datasource 子系统里的 JDBC XA 数据源配置。

[提交 bug 报告](#)

3.5.11. 管理 CLI 操作参考

开放管理 CLI 里的操作

管理 CLI 里的操作可以用 [第 3.6.5 节“用管理 CLI 显示操作名称”](#) 里描述的 **read-operation-names** 操作开放。这些操作描述可以用 [第 3.6.4 节“用管理 CLI 显示操作描述”](#) 里描述的 **read-operation-descriptions** 操作来开放。

表 3.4. 管理 CLI 操作

操作名称	描述
add-namespace	在 <code>namespaces</code> 属性的表里添加命名空间前缀映射。
add-schema-location	在 <code>schema-locations</code> 属性的表里添加模式位置 映射。
delete-snapshot	从 <code>snapshots</code> 目录删除服务器配置的快照。
full-replace-deployment	添加之前上传的部署内容到可用内容列表里，替换 <code>runtime</code> 里具有相同名称的现有内容，并从可用列表里删除替换的内容。进一步的信息请点击链接。
list-snapshots	列出保存在 <code>snapshots</code> 目录里的服务器配置的快照。
read-attribute	显示所选资源的属性的值。
read-children-names	显示给定类型的资源下的所有子资源的名称。
read-children-resources	显示给定类型的资源的所有子资源的信息。
read-children-types	显示所选资源下的所有子资源的类型名称。
read-config-as-xml	读取当前的配置并以 XML 格式显示。
read-operation-description	显示给定资源上的操作的细节。
read-operation-names	显示给定资源上的所有操作的名称。
read-resource	显示模型资源的属性值以及任何子资源的基本或完整的信息。
read-resource-description	显示资源属性的描述、子资源和操作的类型。
reload	关闭所有服务并重启来重载服务器。
remove-namespace	在 <code>namespaces</code> 属性的表里删除命名空间前缀映射。
remove-schema-location	在 <code>schema-locations</code> 属性的表里删除模式位置 映射。
replace-deployment	用新的内容替换 <code>runtime</code> 里的内容。新的内容必须已经上传到部署的内容资料库。
resolve-expression	接受表达式或可以解析为表达式的字符串的操作，并根据本地系统属性和环境变量进行解析。

操作名称	描述
resolve-internet-address	通过一系列接口解析标准找到本地主机上的 IP 地址，如果没有匹配的 IP 地址则运行失败。
server-set-restart-required	让服务器进入需要重启的模式
shutdown	通过调用 System.exit(0) 关闭服务器。
start-servers	启动受管域里配置的且当前没有运行的所有服务器。
stop-servers	停止当前运行在受管域里的所有服务器。
take-snapshot	创建服务器配置的快照并保存在 snapshots 目录。
upload-deployment-bytes	指定所包含的字节队列上的部署内容应该添加到部署内容资料库。请注意，这个操作没有指明内容应该部署至 runtime 。
upload-deployment-stream	指定所包含的输入流索引上可用的部署内容应该添加到部署内容资料库。请注意，这个操作没有指明内容应该部署到 runtime 。
upload-deployment-url	指定所包含的 URL 上可用的部署内容应该添加到部署内容资料库。请注意，这个操作没有指明内容应该部署至 runtime 。
validate-address	检验操作的地址。
write-attribute	设置所选资源的属性的值。

[提交 bug 报告](#)

3.6. 管理 CLI 操作

3.6.1. 用管理 CLI 显示资源属性

必须具备的条件

- [第 3.5.2 节 “启动管理 CLI”](#)

概述

read-attribute 操作是一个用来读取选定属性的当前 **runtime** 值的全局操作。它可以用来只开放用户设置的值而忽略任何默认或未定义的值。请求属性包括下列参数。

请求属性

name

获取所选资源下的值的属性的名称。

include-defaults

布尔型参数，可以设置为 **false** 来限制操作结果，只显示用户设置的属性并忽略默认的值。

过程 3.13. 显示所选属性的当前 Runtime 值

- 运行 **read-attribute** 操作

在管理 CLI 里，使用 **read-attribute** 操作来显示资源属性的值。关于操作请求的更多细节，请参考 [第 3.5.8 节“在管理 CLI 里使用操作和命令”](#)。

```
[standalone@localhost:9999 /]:read-attribute(name=name-of-attribute)
```

read-attribute 操作的优势是开放专有属性的当前 runtime 值的能力。用 **read-resource** 操作可获得类似的结果，但只能通过 **include-runtime** 请求属性，而且只作为该节点的所有可用资源的列表的一部分。**read-attribute** 操作用于细颗粒度的属性查询，如下例所示。

例 3.7. 运行 **read-attribute** 操作来开放公共的接口 IP。

如果您知道要开放的属性的名称，您可以使用 **read-attribute** 来获取当前 runtime 里的确切的值。

```
[standalone@localhost:9999 /] /interface=public:read-attribute(name=resolved-address)
{
  "outcome" => "success",
  "result" => "127.0.0.1"
}
```

resolved-address 属性是一个 runtime 值，所以不会显示在标准的 **read-resource** 操作的结果里。

```
[standalone@localhost:9999 /] /interface=public:read-resource
{
  "outcome" => "success",
  "result" => {
    "any" => undefined,
    "any-address" => undefined,
    "any-ipv4-address" => undefined,
    "any-ipv6-address" => undefined,
    "inet-address" => expression "${jboss.bind.address:127.0.0.1}",
    "link-local-address" => undefined,
    "loopback" => undefined,
    "loopback-address" => undefined,
    "multicast" => undefined,
    "name" => "public",
    "nic" => undefined,
    "nic-match" => undefined,
    "not" => undefined,
    "point-to-point" => undefined,
    "public-address" => undefined,
    "site-local-address" => undefined,
    "subnet-match" => undefined,
    "up" => undefined,
  }
}
```

```

        "virtual" => undefined
    }
}

```

要显示 **resolved-address** 和其他 runtime 值，您必须使用 **include-runtime** 请求属性。

```

[standalone@localhost:9999 /] /interface=public:read-resource(include-
runtime=true)
{
    "outcome" => "success",
    "result" => {
        "any" => undefined,
        "any-address" => undefined,
        "any-ipv4-address" => undefined,
        "any-ipv6-address" => undefined,
        "inet-address" => expression "${jboss.bind.address:127.0.0.1}",
        "link-local-address" => undefined,
        "loopback" => undefined,
        "loopback-address" => undefined,
        "multicast" => undefined,
        "name" => "public",
        "nic" => undefined,
        "nic-match" => undefined,
        "not" => undefined,
        "point-to-point" => undefined,
        "public-address" => undefined,
        "resolved-address" => "127.0.0.1",
        "site-local-address" => undefined,
        "subnet-match" => undefined,
        "up" => undefined,
        "virtual" => undefined
    }
}

```

结果

显示当前的 runtime 属性值。

[提交 bug 报告](#)

3.6.2. 在管理 CLI 里显示活动用户

前提条件

- [第 3.5.2 节“启动管理 CLI”](#)

介绍

whoami 操作是用于确定当前活动用户的全局操作。这个操作开放了用户名标识及它们分配的区。管理员可以用 **whoami** 操作来管理多个区上的多个用户帐号，或者跟踪具有多个终端会话和用户帐号的域实例上的活动用户。

过程 3.14. 在管理 CLI 里用 `whoami` 操作显示活动用户

- 运行 `whoami` 操作

在管理 CLI 里，请用 `whoami` 操作来显示活动的用户帐号。

```
[standalone@localhost:9999 /] :whoami
```

下面的例子使用了独立服务器实例里的 `whoami` 操作来显示活动用户 `username`，以及它所分配的 `ManagementRealm` 区。

例 3.8. 在独立实例里使用 `whoami`

```
[standalone@localhost:9999 /]:whoami
{
  "outcome" => "success",
  "result" => {"identity" => {
    "username" => "username",
    "realm" => "ManagementRealm"
  }}
}
```

结果

显示当前的活动用户帐号。

[提交 bug 报告](#)

3.6.3. 在管理 CLI 里显示系统和服务器信息

前提条件

- [第 3.5.2 节“启动管理 CLI”](#)

过程 3.15. 在管理 CLI 里显示系统和服务器信息

- 运行 `version` 命令

在管理 CLI 里，输入 `version` 命令：

```
[domain@localhost:9999 /] version
```

结果

显示应用服务器版本和环境信息。

[提交 bug 报告](#)

3.6.4. 用管理 CLI 显示操作描述

必须具备的条件

- [第 3.5.2 节“启动管理 CLI”](#)

过程 3.16. 在管理 CLI 里执行命令

- 运行 **read-operation-description** 操作

在管理 CLI 里，使用 **read-operation-description** 来显示操作信息。这个操作要求键-值格式的其他参数以指定要显示的操作。关于操作请求的详情，请参考 [第 3.5.8 节 “在管理 CLI 里使用操作和命令”](#)。

```
[standalone@localhost:9999 /]:read-operation-description(name=name-of-operation)
```

例 3.9. 显示 **list-snapshots** 操作的描述

下面的例子显示了描述 **list-snapshots** 操作的方法。

```
[standalone@localhost:9999 /] :read-operation-description(name=list-snapshots)
{
  "outcome" => "success",
  "result" => {
    "operation-name" => "list-snapshots",
    "description" => "Lists the snapshots",
    "request-properties" => {},
    "reply-properties" => {
      "type" => OBJECT,
      "value-type" => {
        "directory" => {
          "type" => STRING,
          "description" => "The directory where the snapshots
are stored",
          "expressions-allowed" => false,
          "required" => true,
          "nillable" => false,
          "min-length" => 1L,
          "max-length" => 2147483647L
        },
        "names" => {
          "type" => LIST,
          "description" => "The names of the snapshots within
the snapshots directory",
          "expressions-allowed" => false,
          "required" => true,
          "nillable" => false,
          "value-type" => STRING
        }
      }
    },
    "access-constraints" => {"sensitive" => {"snapshots" => {"type"
=> "core"}}},
    "read-only" => false
  }
}
```

结果

显示所选操作的描述。

[提交 bug 报告](#)

3.6.5. 用管理 CLI 显示操作名称

必须具备的条件

- [第 3.5.2 节 “启动管理 CLI”](#)

过程 3.17. 在管理 CLI 里执行命令

- **运行 `read-operation-names` 操作**

在管理 CLI 里，使用 **`read-operation-names`** 来显示可用操作的名称。关于操作请求的详情，请参考 [第 3.5.8 节 “在管理 CLI 里使用操作和命令”](#)。

```
[standalone@localhost:9999 /]:read-operation-names
```

例 3.10. 用管理 CLI 显示操作名称

下面的例子显示了描述 **`read-operation-names`** 操作的方法。

```
[standalone@localhost:9999 /]:read-operation-names
{
  "outcome" => "success",
  "result" => [
    "add-namespace",
    "add-schema-location",
    "delete-snapshot",
    "full-replace-deployment",
    "list-snapshots",
    "read-attribute",
    "read-children-names",
    "read-children-resources",
    "read-children-types",
    "read-config-as-xml",
    "read-operation-description",
    "read-operation-names",
    "read-resource",
    "read-resource-description",
    "reload",
    "remove-namespace",
    "remove-schema-location",
    "replace-deployment",
    "resolve-expression",
    "resolve-internet-address",
    "server-set-restart-required",
    "shutdown",
    "take-snapshot",
    "undefine-attribute",
    "upload-deployment-bytes",
    "upload-deployment-stream",
    "upload-deployment-url",
    "validate-address",
```

```

    "validate-operation",
    "whoami",
    "write-attribute"
  ]
}
```

结果

显示可用的操作名称。

[提交 bug 报告](#)

3.6.6. 用管理 CLI 显示可用资源

必须具备的条件

- [第 3.5.2 节“启动管理 CLI”](#)

概述

read-resource 操作是用来读取资源值的全局操作。它可以用来开放当前节点或子节点额资源的基本或完整的信息，以及扩展或限制操作结果的作用域的请求属性。请求属性包含下列参数。

请求属性

recursive

是否递归地包含子资源的完整信息。

recursive-depth

应该包含子节点资源信息的深度。

proxies

是否在递归查询里包含远程资源。例如，包含域控制器查询里从主机控制器的主机级别资源。

include-runtime

是否在响应里包含 **runtime** 属性，如不是来自持久性配置的属性值。这个请求属性默认是 **false**。

include-defaults

这是一个 **boolean** 型的请求属性，它启用或禁用默认属性的读取。当设置为 **false** 时，只返回用户设置的属性，忽略了那些未定义的属性。

过程 3.18. 在管理 CLI 里执行命令

1. 运行 **read-resource** 操作

在管理 CLI 里，请用 **read-resource** 操作来显示可用的资源。

```
[standalone@localhost:9999 /]:read-resource
```

下面的例子展示了在独立服务器里如何使用 **read-resource** 操作来开放普通资源信息。结果类似于 **standalone.xml** 配置文件，显示系统资源、扩展、接口和为服务器实例安装和配置的系统。它们可以进一步进行直接查询。

例 3.11. 在根级别使用 **read-resource** 操作

```
[standalone@localhost:9999 /]:read-resource
{
  "outcome" => "success",
  "result" => {
    "deployment" => undefined,
    "deployment-overlay" => undefined,
    "management-major-version" => 1,
    "management-micro-version" => 0,
    "management-minor-version" => 4,
    "name" => "longgrass",
    "namespaces" => [],
    "product-name" => "EAP",
    "product-version" => "6.3.0.GA",
    "release-codename" => "Janus",
    "release-version" => "7.2.0.Final-redhat-3",
    "schema-locations" => [],
    "system-property" => undefined,
    "core-service" => {
      "management" => undefined,
      "service-container" => undefined,
      "server-environment" => undefined,
      "platform-mbean" => undefined
    },
    "extension" => {
      "org.jboss.as.clustering.infinispan" => undefined,
      "org.jboss.as.connector" => undefined,
      "org.jboss.as.deployment-scanner" => undefined,
      "org.jboss.as.ee" => undefined,
      "org.jboss.as.ejb3" => undefined,
      "org.jboss.as.jaxrs" => undefined,
      "org.jboss.as.jdr" => undefined,
      "org.jboss.as.jmx" => undefined,
      "org.jboss.as.jpa" => undefined,
      "org.jboss.as.jsf" => undefined,
      "org.jboss.as.logging" => undefined,
      "org.jboss.as.mail" => undefined,
      "org.jboss.as.naming" => undefined,
      "org.jboss.as.pojo" => undefined,
      "org.jboss.as.remoting" => undefined,
      "org.jboss.as.sar" => undefined,
      "org.jboss.as.security" => undefined,
      "org.jboss.as.threads" => undefined,
      "org.jboss.as.transactions" => undefined,
      "org.jboss.as.web" => undefined,
      "org.jboss.as.webservices" => undefined,
      "org.jboss.as.weld" => undefined
    },
    "interface" => {
      "management" => undefined,
      "public" => undefined,

```



```

        "unsecure" => undefined
    },
    "path" => {
        "jboss.server.temp.dir" => undefined,
        "user.home" => undefined,
        "jboss.server.base.dir" => undefined,
        "java.home" => undefined,
        "user.dir" => undefined,
        "jboss.server.data.dir" => undefined,
        "jboss.home.dir" => undefined,
        "jboss.server.log.dir" => undefined,
        "jboss.server.config.dir" => undefined,
        "jboss.controller.temp.dir" => undefined
    },
    "socket-binding-group" => {"standard-sockets" =>
undefined},
    "subsystem" => {
        "logging" => undefined,
        "datasources" => undefined,
        "deployment-scanner" => undefined,
        "ee" => undefined,
        "ejb3" => undefined,
        "infinispan" => undefined,
        "jaxrs" => undefined,
        "jca" => undefined,
        "jdr" => undefined,
        "jmx" => undefined,
        "jpa" => undefined,
        "jsf" => undefined,
        "mail" => undefined,
        "naming" => undefined,
        "pojo" => undefined,
        "remoting" => undefined,
        "resource-adapters" => undefined,
        "sar" => undefined,
        "security" => undefined,
        "threads" => undefined,
        "transactions" => undefined,
        "web" => undefined,
        "webservices" => undefined,
        "weld" => undefined
    }
}
}
}

```

2. 针对子节点运行 read-resource 操作

read-resource 操作可以查询根节点的子节点。操作的结构首先定义要开放的节点，然后附加这个操作来运行。

```
[standalone@localhost:9999 /]/subsystem=web/connector=http:read-
resource
```

在下面的例子里，通过指引 **read-resource** 操作到专有的 **Web** 子系统节点来开放 **Web** 子系统组件的专有资源信息。

例 3.12. 开放根结点的子节点资源

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-resource
{
  "outcome" => "success",
  "result" => {
    "configuration" => undefined,
    "enable-lookups" => false,
    "enabled" => true,
    "executor" => undefined,
    "max-connections" => undefined,
    "max-post-size" => 2097152,
    "max-save-post-size" => 4096,
    "name" => "http",
    "protocol" => "HTTP/1.1",
    "proxy-name" => undefined,
    "proxy-port" => undefined,
    "redirect-port" => 443,
    "scheme" => "http",
    "secure" => false,
    "socket-binding" => "http",
    "ssl" => undefined,
    "virtual-server" => undefined
  }
}
```

相同的结果可以用 **cd** 命令进入子节点并直接运行 **read-resource** 操作获得。

例 3.13. 通过修改目录开放子节点资源

```
[standalone@localhost:9999 /] cd subsystem=web

[standalone@localhost:9999 subsystem=web] cd connector=http

[standalone@localhost:9999 connector=http] :read-resource
{
  "outcome" => "success",
  "result" => {
    "configuration" => undefined,
    "enable-lookups" => false,
    "enabled" => true,
    "executor" => undefined,
    "max-connections" => undefined,
    "max-post-size" => 2097152,
    "max-save-post-size" => 4096,
    "name" => "http",
    "protocol" => "HTTP/1.1",
    "proxy-name" => undefined,
```

```

        "proxy-port" => undefined,
        "redirect-port" => 443,
        "scheme" => "http",
        "secure" => false,
        "socket-binding" => "http",
        "ssl" => undefined,
        "virtual-server" => undefined
    }
}

```

3. 使用 **recursive** 参数在结果里包含活动的属性值。

recursive 参数可以用来开放所有属性的值，包括非持久性的值、在启动时传入的值或其他在 **runtime** 模型里活动的属性。

```

[standalone@localhost:9999 /]/interface=public:read-
resource(include-runtime=true)

```

和之前的例子相比，**include-runtime** 请求属性会开放其他的活动属性，如发送的字节和 HTTP 连接器接收的字节。

例 3.14. 用 **include-runtime** 参数开放其他活动的属性值。

```

[standalone@localhost:9999 /] /subsystem=web/connector=http:read-
resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "any" => undefined,
        "any-address" => undefined,
        "any-ipv4-address" => undefined,
        "any-ipv6-address" => undefined,
        "inet-address" => expression
        "${jboss.bind.address:127.0.0.1}",
        "link-local-address" => undefined,
        "loopback" => undefined,
        "loopback-address" => undefined,
        "multicast" => undefined,
        "name" => "public",
        "nic" => undefined,
        "nic-match" => undefined,
        "not" => undefined,
        "point-to-point" => undefined,
        "public-address" => undefined,
        "resolved-address" => "127.0.0.1",
        "site-local-address" => undefined,
        "subnet-match" => undefined,
        "up" => undefined,
        "virtual" => undefined
    }
}

```

[提交 bug 报告](#)

3.6.7. 用管理 CLI 显示可用资源的描述

必须具备的条件

- [第 3.5.2 节 “启动管理 CLI”](#)

过程 3.19. 在管理 CLI 里执行命令

1. 运行 `read-resource-description` 操作

在管理 CLI 里，使用 `read-resource-description` 来读取和显示可用资源。关于操作请求的详情，请参考 [第 3.5.8 节 “在管理 CLI 里使用操作和命令”](#)。

```
[standalone@localhost:9999 /]:read-resource-description
```

2. 使用可选参数

`read-resource-description` 操作允许使用其他参数。

- a. 使用 `operations` 参数来包含资源操作的描述。

```
[standalone@localhost:9999 /]:read-resource-  
description(operations=true)
```

- b. 使用 `inherited` 参数可以包含或排除资源继承操作的描述。默认状态是 `true`。

```
[standalone@localhost:9999 /]:read-resource-  
description(inherited=false)
```

- c. 使用 `recursive` 参数来包含子资源的递归描述。

```
[standalone@localhost:9999 /]:read-resource-  
description(recursive=true)
```

- d. 使用 `locale` 参数来获取资源描述。如果为 `null` 则使用默认的 `locale`。

```
[standalone@localhost:9999 /]:read-resource-  
description(locale=true)
```

结果

显示可用资源的描述。

[提交 bug 报告](#)

3.6.8. 用管理 CLI 重载应用服务器

前提条件

- [第 3.5.2 节 “启动管理 CLI”](#)

在管理 CLI 里使用 **reload** 操作来关闭所有服务并重启 runtime。在完成 **reload** 操作后管理 CLI 将自动重连。

关于操作请求的更多细节，请参考 [第 3.5.8 节“在管理 CLI 里使用操作和命令”](#)。

例 3.15. 重载应用服务器

```
[standalone@localhost:9999 /]:reload
{"outcome" => "success"}
```

[提交 bug 报告](#)

3.6.9. 用管理 CLI 关闭应用服务器

前提条件

- [第 3.5.2 节“启动管理 CLI”](#)

过程 3.20. 关闭应用服务器

- 运行 **shutdown** 操作

- 在管理 CLI 里，使用 **shutdown** 操作通过 **System.exit(0)** 系统调用来关闭服务器。关于操作请求的详情，请参考 [第 3.5.8 节“在管理 CLI 里使用操作和命令”](#)。

- 在独立模式下，请使用下列命令：

```
[standalone@localhost:9999 /]:shutdown
```

- 在域模式下，请使用下列命令及合适的主机名：

```
[domain@localhost:9999 /]/host=master:shutdown
```

- 要连接到附加的 CLI 实例并关闭服务器，请执行下列命令：

```
jboss-cli.sh --connect command=:shutdown
```

- 要连接到远程的 CLI 实例并关闭服务器，请执行下列命令：

```
[disconnected /] connect IP_ADDRESS
Connected to IP_ADDRESS:PORT_NUMBER
[192.168.1.10:9999 /] :shutdown
```

用实例的 IP 地址替换 *IP_ADDRESS*

结果

服务器被关闭。管理 CLI 将断开连接，因为 runtime 已是不可用的。

[提交 bug 报告](#)

3.6.10. 使用管理 CLI 配置属性

前提条件

- [第 3.5.2 节 “启动管理 CLI”](#)

介绍

write-attribute 操作是用来写入或修改资源属性的全局操作。您可以使用这个操作来进行持久性修改并修改管理的服务器实例的配置设置。请求属性包含下列参数。

请求属性

name

需要设置值的所选资源属性的名称。

value

所选资源里属性的值。如果底层模型支持 null 值的话可以为 null。

过程 3.21. 使用管理 CLI 配置资源属性

- **运行 write-attribute 操作**

在管理 CLI 里，使用 **write-attribute** 操作修改资源属性的值。这个操作可以运行在资源的子节点或管理 CLI 的根节点上（指定完整资源路径）。

例 3.16. 用 write-attribute 操作禁用部署扫描器。

下面的例子使用了 **write-attribute** 操作来禁用部署扫描器。这个操作从根节点运行，使用 **Tab Completion** 来协助填充正确的资源路径。

```
[standalone@localhost:9999 /] /subsystem=deployment-
scanner/scanner=default:write-attribute(name=scan-enabled,value=false)
{"outcome" => "success"}
```

操作的结果可以直接用 **read-attribute** 操作确认。

```
[standalone@localhost:9999 /] /subsystem=deployment-
scanner/scanner=default:read-attribute(name=scan-enabled)
{
  "outcome" => "success",
  "result" => false
}
```

用 **read-resource** 操作列出节点的所有可用资源属性可以确认资源。在下列例子里，这个特定的配置展示了 **scan-enabled** 属性被设置为 **false**。

```
[standalone@localhost:9999 /] /subsystem=deployment-
scanner/scanner=default:read-resource
{
  "outcome" => "success",
```

```

    "result" => {
      "auto-deploy-exploded" => false,
      "auto-deploy-xml" => true,
      "auto-deploy-zipped" => true,
      "deployment-timeout" => 600,
      "path" => "deployments",
      "relative-to" => "jboss.server.base.dir",
      "scan-enabled" => false,
      "scan-interval" => 5000
    }
  }
}

```

结果

更新了资源属性

[提交 bug 报告](#)

3.6.11. 用管理 CLI 配置系统属性

过程 3.22. 用管理 CLI 配置系统属性

1. 启动 JBoss EAP 服务器。
2. 使用适合操作系统的命令启动管理 CLI

对于 Linux :

```
EAP_HOME/bin/jboss-cli.sh --connect
```

对于 Windows:

```
EAP_HOME\bin\jboss-cli.bat --connect
```

3. 添加系统属性。

使用的命令取决于服务器是否是独立服务器还是运行在受管域里。如果您运行的是受管域，您可以添加任何系统属性到运行在这个域里的任何或所有服务器里。

- 用下列语法在独立服务器上添加一个系统属性：

```
/system-property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

例 3.17. 添加系统属性到独立服务器

```

[standalone@localhost:9999 /] /system-
property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)
{"outcome" => "success"}

```

- 用下列语法在受管域里的所有主机和服务器上添加一个系统属性：

```
/system-property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

例 3.18. 添加系统属性到受管域里的所有服务器

```
[domain@localhost:9999 /] /system-
property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" =>
{"master" => {
    "server-one" => {"response" => {"outcome" =>
"success"}}},
    "server-two" => {"response" => {"outcome" =>
"success"}}
    }}}}
```

- 用下列语法在受管域里的主机及其服务器实例上添加一个系统属性：

```
/host=master/system-
property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

例 3.19. 添加系统属性到域里的主机及其服务器

```
[domain@localhost:9999 /] /host=master/system-
property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" =>
{"master" => {
    "server-one" => {"response" => {"outcome" =>
"success"}}},
    "server-two" => {"response" => {"outcome" =>
"success"}}
    }}}}
```

- 用下列语法添加一个系统属性到受管域里的服务器实例：

```
/host=master/server-config=server-one/system-
property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

例 3.20. 添加系统属性到受管域里的服务器实例

```
[domain@localhost:9999 /] /host=master/server-config=server-
one/system-
```



```

property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" =>
{"master" => {"server-one" => {"response" => {"outcome" =>
"success"}}}}}}}}
}

```

4. 读取系统属性。

所使用的命令取决于服务器是否是独立服务器还是运行在受管域里。

- 用下列语法从独立服务器读取系统属性：

```
/system-property=PROPERTY_NAME:read-resource
```

例 3.21. 从独立服务器读取系统属性

```

[standalone@localhost:9999 /] /system-
property=property.mybean.queue:read-resource
{
    "outcome" => "success",
    "result" => {"value" => "java:/queue/MyBeanQueue"}
}

```

- 用下列语法在受管域里的所有主机和服务器上读取系统属性：

```
/system-property=PROPERTY_NAME:read-resource
```

例 3.22. 从受管域里的所有服务器上读取系统属性

```

[domain@localhost:9999 /] /system-
property=property.mybean.queue:read-resource
{
    "outcome" => "success",
    "result" => {
        "boot-time" => true,
        "value" => "java:/queue/MyBeanQueue"
    }
}

```

- 用下列语法从受管域里的主机及其服务器实例里读取系统属性：

```
/host=master/system-property=PROPERTY_NAME:read-resource
```

例 3.23. 从受管域里的主机及其服务器读取系统属性

```
[domain@localhost:9999 /] /host=master/system-
property=property.mybean.queue:read-resource
{
    "outcome" => "success",
    "result" => {
        "boot-time" => true,
        "value" => "java:/queue/MyBeanQueue"
    }
}
```

- 用下列语法从受管域里的服务器实例里读取系统属性：

```
/host=master/server-config=server-one/system-
property=PROPERTY_NAME:read-resource
```

例 3.24. 从受管域里的服务器实例读取系统属性

```
[domain@localhost:9999 /] /host=master/server-config=server-
one/system-property=property.mybean.queue:read-resource
{
    "outcome" => "success",
    "result" => {
        "boot-time" => true,
        "value" => "java:/queue/MyBeanQueue"
    }
}
```

5. 删除系统属性。

所使用的命令取决于服务器是否是独立服务器还是运行在受管域里。

- 用下列语法从独立服务器删除系统属性：

```
/system-property=PROPERTY_NAME:remove
```

例 3.25. 从独立服务器删除系统属性

```
[standalone@localhost:9999 /] /system-
property=property.mybean.queue:remove
{"outcome" => "success"}
```

- 用下列语法从受管域里的所有主机和服务器里删除系统属性：

```
/system-property=PROPERTY_NAME:remove
```

例 3.26. 从受管域里的所有主机和服务器里删除系统属性

```
[domain@localhost:9999 /] /system-
property=property.mybean.queue:remove
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" =>
{"master" => {
    "server-one" => {"response" => {"outcome" =>
"success"}}},
    "server-two" => {"response" => {"outcome" =>
"success"}}
}}}
}
```

- 用下列语法从受管域里的主机及其服务器实例里删除系统属性：

```
/host=master/system-property=PROPERTY_NAME:remove
```

例 3.27. 从受管域里的主机及其实例里删除系统属性

```
[domain@localhost:9999 /] /host=master/system-
property=property.mybean.queue:remove
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" =>
{"master" => {
    "server-one" => {"response" => {"outcome" =>
"success"}}},
    "server-two" => {"response" => {"outcome" =>
"success"}}
}}}
}
```

- 用下列语法从受管域里的服务器实例里删除系统属性：

```
/host=master/server-config=server-one/system-
property=PROPERTY_NAME:remove
```

例 3.28. 从受管域里的服务器里删除系统属性

```
[domain@localhost:9999 /] /host=master/server-config=server-
one/system-property=property.mybean.queue:remove
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" =>
{"master" => {"server-one" => {"response" => {"outcome" =>
"success"}}}}
}
```

```
"success"}}}}}}
}
```

[提交 bug 报告](#)

3.7. 管理 CLI 命令历史

3.7.1. 使用管理 CLI 命令历史

应用服务器安装时会默认启用管理 CLI 的历史命令功能。这个历史记录既在活动的 CLI 会话的易变内存里保持一条记录，也附加内容在自动保存在用户的主目录的 **.jboss-cli-history** 日志文件上。这个历史记录文件默认是记录最多 500 条 CLI 命令。

history 命令自身将返回当前会话的历史记录，或用其他参数将禁用、启用或清除会话内存里的历史记录。管理 CLI 也可以通过键盘上的箭头来在命令和操作的历史记录里前进或后退。

管理 CLI 的 history 命令的功能

- [第 3.7.2 节 “显示管理 CLI 命令历史”](#)
- [第 3.7.3 节 “清除管理 CLI 命令历史”](#)
- [第 3.7.4 节 “禁用管理 CLI 命令历史”](#)
- [第 3.7.5 节 “启用管理 CLI 命令历史”](#)

[提交 bug 报告](#)

3.7.2. 显示管理 CLI 命令历史

前提条件

- [第 3.5.2 节 “启动管理 CLI”](#)

过程 3.23. 显示管理 CLI 命令历史

- **运行 history 命令**
在管理 CLI 里，输入 **history** 命令：

```
[standalone@localhost:9999 /] history
```

结果

在 CLI 启动或历史清除命令显示后保存在内存里的 CLI 命令历史。

[提交 bug 报告](#)

3.7.3. 清除管理 CLI 命令历史

前提条件

- [第 3.5.2 节 “启动管理 CLI”](#)

过程 3.24. 清除管理 CLI 命令历史

- **运行 `history --clear` 命令**
在管理 CLI 里，输入 `history --clear` 命令：

```
[standalone@localhost:9999 /] history --clear
```

结果

自 CLI 启动后记录的命令历史将从会话内存里删除。这些命令历史仍然保存在用户主目录的 `.jboss-cli-history` 文件里。

[提交 bug 报告](#)

3.7.4. 禁用管理 CLI 命令历史

前提条件

- [第 3.5.2 节 “启动管理 CLI”](#)

过程 3.25. 禁用管理 CLI 命令历史

- **运行 `history --disable` 命令**
在管理 CLI 里，输入 `history --disable` 命令：

```
[standalone@localhost:9999 /] history --disable
```

结果

CLI 里执行的命令不会记录在内存里或保存在用户主目录的 `.jboss-cli-history` 文件里。

[提交 bug 报告](#)

3.7.5. 启用管理 CLI 命令历史

前提条件

- [第 3.5.2 节 “启动管理 CLI”](#)

过程 3.26. 启用管理 CLI 命令历史

- **运行 `history --enable` 命令**
在管理 CLI 里，输入 `history --enable` 命令：

```
[standalone@localhost:9999 /] history --enable
```

结果

CLI 里执行的命令会记录在内存里并保存在用户主目录的 `.jboss-cli-history` 文件里。

[提交 bug 报告](#)

3.8. 管理接口审计日志

3.8.1. 关于管理接口审计日志

启用审计日志后，通过管理 CLI 执行的操作将在审计日志记录。不管这些操作是否通过管理控制台、管理 CLI 还是自定义的接口执行，都会登记审计日志。日志记录可以输出到文件里或转发到 Syslog 服务器，也可以两者都进行。在默认情况下，审计日志是禁用的。

日志数据以 JSON 格式输出，并有几个配置选项可以影响日志里的操作及日志条目的格式。

在存储日志之前，格式器和处理程序会先处理日志条目。格式器指定日志条目的格式，而处理程序输出记录到指定的目的地。目前只有一个可用的格式器，它以 JSON 格式输出条目。



注意

审计日志只能通过管理 CLI 进行配置。

[提交 bug 报告](#)

3.8.2. 从管理 CLI 里启用管理接口的审计日志

要在管理 CLI 启用审计日志，请使用下列命令。

```
/core-service=management/access=audit/logger=audit-log:write-attribute(name=enabled,value=true)
```

审计日志是预配置输出到 `EAP_HOME/standalone/data/audit-log.log` 里的。

[提交 bug 报告](#)

3.8.3. 关于管理接口的审计日志格式器

格式器指定日志条目的格式。

表 3.5. JSON 格式器字段

属性	描述
include-date	布尔值，它定义格式化日志记录是否包含时间戳。
date-separator	包含分隔日期和格式化日志信息的字符的字符串。如果 include-date=false 则被忽略。
date-format	用于时间戳的 <code>java.text.SimpleDateFormat</code> 。可以时别的日期格式。如果 include-date=false 则被忽略。
compact	如果为 true ，它将在一行里格式化 JSON 信息。因为仍有包含新行符的值，所以如果需要在同一行里容纳整个记录，可以设置 escape-new-line 或 escape-control-characters 为 true 。

属性	描述
<code>escape-control-characters</code>	如果为 true ，它将所有带有八进制 ASCII 字符的控制字符（带有十进制值 < 32 的 ASCII 条目）转义；例如新行将转义为 '#012'。如果它为 true ，它将覆盖 <i>escape-new-line=false</i> 。
<code>escape-new-line</code>	如果为 true 会将所有带有八进制的 ASCII 代码的新行转义，例如 #012 。

[提交 bug 报告](#)

3.8.4. 关于管理接口的审计日志文件处理程序

文件处理程序通过参数来指定哪些日志记录输出到文件里。它还定义格式器、文件名和路径。

表 3.6. 文件处理程序的审计日志字段

属性	描述	只读的
<code>formatter</code>	用来格式化日志记录的 JSON 格式器的名称。	False
<code>path</code>	审计日志文件的路径。	False
<code>relative-to</code>	之前的命名路径的名称，或者系统提供的标准路径中的一个。如果提供了 <i>relative-to</i> ， <code>path</code> 属性的值将作为这个属性指定的路径的相对路径对待。	False
<code>failure-count</code>	初始化处理程序前记录日志失败的次数。	True
<code>max-failure-count</code>	禁用这个处理程序前记录日志失败的最多次数。	False
<code>disabled-due-to-failure</code>	true 表示如果登记日志失败则禁用处理程序。	True

[提交 bug 报告](#)

3.8.5. 关于管理接口的审计日志 Syslog 处理程序

Syslog 处理程序通过参数指定哪些审计日志条目被发送到 Syslog 服务器，特别是 Syslog 服务器侦听的主机名和端口。

发送审计日志到 Syslog 服务器比本地文件或本地 Syslog 服务器提供了更安全的选项。您可以定义多个 Syslog 处理程序。

Syslog 服务器有不同的实现，所以并非所有设置都可应用到所有的 Syslog 服务器上。我们已用 *rsyslog* syslog 实现进行了测试。引用的 RFC 是：

- <http://www.ietf.org/rfc/rfc3164.txt>
- <http://www.ietf.org/rfc/rfc5424.txt>
- <http://www.ietf.org/rfc/rfc6587.txt>

表 3.7. Syslog 处理程序字段

字段	描述	只读的值
formatter	用来格式化日志记录的格式器的名称。	False
failure-count	初始化处理程序前记录日志失败的次数。	True
max-failure-count	禁用这个处理程序前记录日志失败的最多次数。	False
disabled-due-to-failure	True 表示如果登记日志失败则禁用处理程序。	True
syslog-format	Syslog 格式： <i>RFC-5424</i> 或 <i>RFC-3164</i> 。	False
max-length	日志消息的最大长度（字节），包括头部信息。如果没有定义， syslog-format 为 RFC3164 时它默认为 1024 字节，而 syslog-format 为 RFC5424 时它默认为 2048 字节。	False.
truncate	如果消息长度超过最大字节数，是否要截短（包括头部信息）。如果为 false ，消息将即兴分隔然后用相同的头部值来发送。	False

[提交 bug 报告](#)

3.8.6. 启用发送到 Syslog 服务器的管理接口审计日志



注意

如果要应用修改到受管域，请添加前缀 `/host=HOST_NAME` 到 `/core-service` 命令上。

过程 3.27. 启用发送到 Syslog 服务器的日志

1. 创建一个名为 `mysyslog` 的 `syslog` 处理程序

```
[standalone@localhost:9999 /]batch
[standalone@localhost:9999 /]/core-
service=management/access=audit/syslog-
handler=mysyslog:add(formatter=json-formatter)
[standalone@localhost:9999 /]/core-
service=management/access=audit/syslog-
handler=mysyslog/protocol=udp:add(host=localhost,port=514)
[standalone@localhost:9999 /]run-batch
```

2. 在 `syslog` 处理程序里添加一个引用。

```
[standalone@localhost:9999 /]/core-
service=management/access=audit/logger=audit-
log/handler=mysyslog:add
```

结果

管理接口的审计日志登记在 `syslog` 服务器上。

[提交 bug 报告](#)

3.8.7. 管理接口的审计日志选项

除了启用和禁用管理接口审计日志，还有其他可用的配置选项。

配置选项

`log-boot`

如果为 `true`，引导服务器时的管理操作将记录审计日志。如果为 `false` 则不会记录日志。默认为 `false`。

`log-read-only`

如果为 `true`，所有的操作都将记录审计日志。如果为 `false`，只有修改了模型的操作会记录日志。默认为 `false`。

[提交 bug 报告](#)

3.8.8. 管理接口的审计日志字段

表 3.8. 管理接口的审计日志字段

字段名称	描述
type	它的值如果为 <code>core</code> ，表示是一个管理操作；如果为 <code>jmx</code> ，表示它来自 JMX 在系统（关于 JMX 子系统的审计日志请参考 JMX 子系统部分）。
r/o	如果操作没有修改管理模型则为 <code>true</code> ，否则为 <code>false</code> 。

字段名称	描述
booting	如果操作是在引导过程中执行的则为 true ，如果是在服务器启动并运行后执行的则为 false 。
version	JBoss EAP 实例的版本号码。
user	已验证用户的用户名。如果和运行的服务器相同的主机为这个操作登记了日志，它将使用特殊的 \$local 用户。
domainUUID	当所有操作从域控制器传播到服务器、从主机控制器和从主机控制器服务器时，链接所有操作的标识符。
access	它可以是下列值之一：NATIVE、HTTP、JMX。 NATIVE - 操作通过原生管理接口进行，如 CLI。HTTP - 操作通过域 HTTP 接口进行，例如域控制台。JMX - 操作通过 JMX 子系统进行。关于如何配置 JMX 的审计日志，请参考 JMX 文档。
remote-address	执行这个操作的客户的地址。
success	如果操作成功则为 true ，如果回滚则为 false 。
ops	被执行的操作。这是一个序列化到 JSON 的操作的列表。在引导时这是解析 XML 导致的所有操作。引导完成后，这个列表通常只包含单个条目。

[提交 bug 报告](#)

第 4 章 用户管理

4.1. 用户创建

4.1.1. 为管理接口添加用户

介绍

JBoss EAP 6 里的管理界面默认是设置了安全性的，因为一开始没有可用的用户帐号，除非你是用图形安装程序安装的。对于因简单配置错误而可能引起来自远程系统的攻击来说，这是一个预防措施。本地的非 HTTP 访问是受 SASL 机制保护的，就是当客户从 `localhost` 第一次连接时客户和服务器间都进行协商。

这个任务描述了如何创建初始的管理性用户，它可以使用基于 WEB 的管理控制台和管理 CLI 的远程实例来从远程系统上配置和管理 JBoss EAP 6。



注意

和 JBoss EAP 6 的 HTTP 通讯被当作是远程访问，即使这种通讯发生在本地主机。因此，你必须创建至少一个用户以能够使用管理控制台。如果你试图在添加用户前访问管理控制台，你将接收到一个错误，因为它在用户创建后才会被部署。

过程 4.1. 为远程管理界面创建初始管理性用户

1. 调用 `add-user.sh` 或 `add-user.bat` 脚本。

进入 `EAP_HOME/bin/` 目录。根据你的操作系统调用合适的脚本。

Red Hat Enterprise Linux

```
[user@host bin]$ ./add-user.sh
```

Microsoft Windows Server

```
C:\bin> add-user.bat
```

2. 选择添加一个管理用户。

点击 **ENTER** 选择默认选项 **a** 来添加一个管理用户。这个用户被添加到 **ManagementRealm** 并被授权通过基于 **web** 的管理控制台或基于命令行的管理 **CLI** 来执行管理操作。另外一个选项 **b** 则添加一个用户到 **ApplicationRealm**，且未提供特殊的权限。该区域 (**Realm**) 用于应用程序。

3. 输入用户名和密码。

遇到提示时输入用户名和密码，系统会提示您确认密码。

4. 输入您的组信息

添加用户所属的组或组群。如果用户属于多个组，请输入用逗号隔开的列表。如果不属于任何组，请留空。

5. 获取信息并确认。

系统会提示您确认信息。如果正确，请输入 **yes**。

6. 选择用户是否代表一个远程 JBoss EAP 6 服务器实例。

除了管理员以外，偶尔需要在 **ManagementRealm** 里添加到 JBoss EAP 6 里的是代表其他 EAP 实例的用户，它需要通过验证作为成员加入到群集。下一个提示允许你指定所添加的用户。如果

你选择 **yes**，你将得到一个 **hashed secret** 值，代表用户的密码，这将需要添加到不同的配置文件里。为了完成这个任务，在这里请回答 **no**。

7. 输入其他的用户。

如果需要，重复刚才的过程你可以输入其他用户。你也可以在任何时候在运行系统里添加用户。不是选择默认的安全区，你需要添加用户到其他区以调整其授权过程。

8. 非交互式地创建用户。

通过在命令行传入参数，你可以非交互式地创建用户。我们不推荐在共享系统上使用这个方法，因为密码可以在日志或历史文件里看到。这个使用管理区域的命令的语法是：

```
[user@host bin]$ ./add-user.sh username password
```

要使用应用程序区域，请使用 **-a** 参数。

```
[user@host bin]$ ./add-user.sh -a username password
```

9. 通过 **--silent** 参数，你可以忽略 **add-user** 脚本的正常输出。这只有在指定了 **用户名和密码**且只使用了最小参数集时才适用。而错误信息仍会被显示。

结果

你添加的任何用户都会在你指定的安全区里进行激活。**ManagementRealm** 区里活动的用户能够从远程系统里管理 JBoss EAP 6。

还可查看：

- [第 11.8.1 节 “默认的用户安全性配置”](#)

[提交 bug 报告](#)

4.1.2. 传入参数到用户管理 **add-user** 脚本

您可以交互式地运行 **add-user.sh** 或 **add-user.bat** 命令或将参数传入到命令行。本节描述了传入参数时可用的选项。

关于 **add-user.sh** 或 **add-user.bat** 命令行的完整参数列表，请参考 [第 4.1.3 节 “Add-user 命令行参数”](#)。

关于如何指定其他属性文件和位置，请参考 [第 4.1.4 节 “指定用户管理信息的替代属性文件”](#)。

关于演示如何传递参数到 **add-user.sh** 或 **add-user.bat** 命令行的示例，请参考 [第 4.1.5 节 “Add-user 脚本命令行示例”](#)。

[提交 bug 报告](#)

4.1.3. Add-user 命令行参数

下表描述了 **add-user.sh** 或 **add-user.bat** 命令的可用参数。

表 4.1. Add-user 命令行参数

命令行参数	参数值	描述
-a	N/A	这个参数指定在应用程序区里创建用户。如果忽略，默认是在管理区里创建用户。
-dc	<i>DOMAIN_CONFIGURATION_DIRECTORY</i>	这个参数指定了包含属性文件的域配置目录。如果忽略，默认的目录是 EAP_HOME/domain/configuration/ 。
-sc	<i>SERVER_CONFIGURATION_DIRECTORY</i>	这个参数指定了包含属性文件的其他独立服务器配置目录。如果忽略，默认的目录是 EAP_HOME/standalone/configuration/ 。
-up --user-properties	<i>USER_PROPERTIES_FILE</i>	这个参数指定其他用户属性文件的名称。它可以是一个绝对路径，也可以和 -sc 或 -dc 参数一起来指定其他配置的目录。
-g --group	<i>GROUP_LIST</i>	分配给这个用户的用逗号隔开的组的列表。
-gp --group-properties	<i>GROUP_PROPERTIES_FILE</i>	这个参数指定其他组属性文件的名称。它可以是一个绝对路径，也可以和 -sc 或 -dc 参数一起来指定其他配置的目录。
-p --password	<i>PASSWORD</i>	用户的密码。密码必须满足下列要求： <ul style="list-style-type: none"> • 它必须包含至少 8 个字符。 • 它必须包含至少一个字母字符。 • 它必须包含至少一个数字。 • 它必须包含至少一个非字母数字字符。
-u --user	<i>USER_NAME</i>	用户的名称。它必须只包含字母数字字符。
-r --realm	<i>REALM_NAME</i>	用来设置管理接口安全性的区的名称。如果忽略，默认是 ManagementRealm 。
-s --silent	N/A	运行 add-user 脚本且不输出到控制台。
-h --help	N/A	显示 add-user 脚本的用法。

4.1.4. 指定用户管理信息的替代属性文件

介绍

在默认情况下，用 **add-user.sh** 或 **add-user.bat** 创建的用户和角色信息都保存在服务器配置目录下的属性文件里。服务器配置信息保存在 **EAP_HOME/standalone/configuration/** 目录而域配置信息保存在 **EAP_HOME/domain/configuration/** 目录。本节将描述如何覆盖默认的文件名称和位置。

过程 4.2. 指定替代属性文件

- 要指定服务器配置的替代目录，请使用 **-sc** 参数。这个参数指定了包含服务器配置属性文件的替代目录。
- 要为域配置指定替代目录，，请使用 **-dc** 参数。这个参数指定了包含域配置属性文件的替代目录。
- 要指定替代的用户配置属性文件，请使用 **-up** 或 **--user-properties** 参数。它可以是绝对路径，也可以和 **-sc** 或 **-dc** 参数一起使用来指定替代的配置目录。
- 要指定替代的组配置属性文件，请使用 **-gp** 或 **--group-properties** 参数。它可以是绝对路径，也可以和 **-sc** 或 **-dc** 参数一起使用来指定替代的配置目录。



注意

add-user 命令旨在操作现有的属性文件。命令行里指定任何替代属性文件都必须存在，否则您将看到下列错误：

```
JBAS015234: No appusers.properties files found
```

关于命令参数的更多信息，请参考 [第 4.1.3 节 “Add-user 命令行参数”](#)。

关于 **add-user** 命令的示例，请参考 [第 4.1.5 节 “Add-user 脚本命令行示例”](#)。

[提交 bug 报告](#)

4.1.5. Add-user 脚本命令行示例

下面的例子演示了如何传递参数到 **add-user.sh** 或 **add-user.bat** 命令。除非另有注明，这些命令假定是使用独立服务器配置的。

例 4.1. 创建属于使用默认属性文件的单个组的用户。

```
EAP_HOME/bin/add-user.sh -a -u 'appuser1' -p 'password1!' -g 'guest'
```

上面的命令产生下列结果。

- 用户 **appuser1** 被添加到保存用户信息的下列默认属性文件里。
EAP_HOME/standalone/configuration/application-users.properties
EAP_HOME/domain/configuration/application-users.properties
- 用户 **appuser1** 和组 **guest** 被添加到保存组信息的默认属性文件里。

`EAP_HOME/standalone/configuration/application-roles.properties`

`EAP_HOME/domain/configuration/application-roles.properties`

例 4.2. 创建属于使用默认属性文件的多个组的用户。

```
EAP_HOME/bin/add-user.sh -a -u 'appuser1' -p 'password1!' -g
'guest,app1group,app2group'
```

上面的命令产生下列结果。

- 用户 **appuser1** 被添加到保存用户信息的下列默认属性文件里。

`EAP_HOME/standalone/configuration/application-users.properties`

`EAP_HOME/domain/configuration/application-users.properties`

- 用户 **appuser1** 和组 **guest**、**app1group** 和 **app2group** 被添加到保存组信息的默认属性文件里。

`EAP_HOME/standalone/configuration/application-roles.properties`

`EAP_HOME/domain/configuration/application-roles.properties`

例 4.3. 在使用默认属性文件的默认区里创建带有管理权限的用户。

```
EAP_HOME/bin/add-user.sh -u 'adminuser1' -p 'password1!' -g 'admin'
```

上面的命令产生下列结果。

- 用户 **adminuser1** 被添加到保存用户信息的下列默认属性文件里。

`EAP_HOME/standalone/configuration/mgmt-users.properties`

`EAP_HOME/domain/configuration/mgmt-users.properties`

- 用户 **adminuser1** 和组 **admin** 被添加到保存组信息的默认属性文件里。

`EAP_HOME/standalone/configuration/mgmt-groups.properties`

`EAP_HOME/domain/configuration/mgmt-groups.properties`

例 4.4. 创建属于用替代属性文件保存信息的单个组的用户。

```
EAP_HOME/bin/add-user.sh -a -u appuser1 -p password1! -g app1group -sc
/home/someusername/userconfigs/ -up appusers.properties -gp
appgroups.properties
```

上面的命令产生下列结果。

- 我们添加了用户 **appuser1** 到下列属性文件里，它现在是保存用户信息的默认文件。

/home/someusername/userconfigs/appusers.properties

- 我们添加了用户 **appuser1** 和组 **app1group** 到下列属性文件里，它现在是保存组信息的默认文件。

/home/someusername/userconfigs/appgroups.properties

[提交 bug 报告](#)

第 5 章 网络和端口配置

5.1. 接口

5.1.1. 关于接口

应用服务器在整个配置里都使用命名接口引用。这让配置可以用逻辑名称引用单独的接口声明，而不是每次都使用整个接口细节。逻辑名称的使用也保持了组引用和命名接口的一致性，而受管域上的服务器实例可能包含多个主机上不同的接口细节。使用这个方法，每个服务器实例可能对应逻辑名称组，它可以将接口组作为整体进行管理。

网络接口是通过指定逻辑名和物理接口的选择标准来声明的。应用服务器附带有用于管理及公共接口名称的默认配置。在这个配置里，公共接口组的目的是用于和应用程序相关的网络通讯，如 **Web** 或 **Messaging**。管理接口组的目的是用于管理层要求的所有组件和服务，包括 **HTTP** 管理端点。接口名自身是仅作为建议来提供的，任何组的名称都可以按照需要进行替换和创建。

domain.xml, **host.xml** 和 **standalone.xml** 都包含了接口声明。声明标准可以引用通配符地址或指定接口或地址必须具有来进行有效匹配的一个或多个特征。下面的例子展示了接口声明的多个可能的配置，它们通常是在 **standalone.xml** 或 **host.xml** 配置文件里定义的。这允许任何远程主机组维护自己所专有的接口属性，且仍然允许对域控制器里的 **domain.xml** 配置文件里任何接口组的引用。

第一个例子展示了为 **management** 和 **public** 相对名称组指定的专有的 **inet-address** 值。

例 5.1. 用 **inet-address** 值创建的接口组

```
<interfaces>
  <interface name="management">
    <inet-address value="127.0.0.1"/>
  </interface>
  <interface name="public">
    <inet-address value="127.0.0.1"/>
  </interface>
</interfaces>
```

在下面的例子里，全局接口组使用了 **any-address** 元素来声明通配符地址。

例 5.2. 用通配符声明创建的全局组

```
<interface name="global">
  <!-- Use the wild-card address -->
  <any-address/>
</interface>
```

下面的例子声明了名为 **external** 的相对组下的一个网络接口卡。

例 5.3. 用 **NIC** 值创建的外部组

```
<interface name="external">
  <nic name="eth0"/>
</interface>
```

在下面的例子里，根据专有需求，声明作为默认组创建。在这个实例里，其他元素的特征为接口设置了条件以进行有效的匹配。这允许每个专有接口声明组的创建，且能够以当前模式引用它们，从而减少了跨多个服务器实例的配置和管理。

例 5.4. 用专有条件值创建的默认组

```
<interface name="default">
  <!-- Match any interface/address on the right subnet if it's
        up, supports multicast, and isn't point-to-point -->
  <subnet-match value="192.168.0.0/16"/>
  <up/>
  <multicast/>
  <not>
    <point-to-point/>
  </not>
</interface>
```

虽然接口声明可以在源配置文件里创建和编辑，管理 CLI 和管理控制台为配置的修改提供了安全的、可控和持久性的环境。

[提交 bug 报告](#)

5.1.2. 配置接口

standalone.xml 和 **host.xml** 配置文件里默认的接口配置通常提供三个带有相对接口令牌的命名接口。您可以使用管理控制台或管理 CLI 来配置下表列出的其他属性和值。你也可以按需要用专有值替换相对的接口绑定。请注意，如果您这样做，您将无法在运行时传入接口值，因为 **-b** 选项只能覆盖相对值。

例 5.5. 默认的接口配置

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

表 5.1. 接口属性和值

接口元素	描述
any	地址排斥类型的空元素，用于约束选择标准。
any-address	空元素表示使用这个接口的套接字应该绑定到通配符地址。除非设置 <code>java.net.preferIPv4Stack</code> 系统属性为 <code>true</code> ，否则 IPv6 将使用通配符地址 (::)，而 IPv4 将使用通配符地址 (0.0.0.0)。如果套接字绑定到双栈主机上的任何本地 IPv6 地址，它可以接受 IPv6 及 IPv4 数据。如果绑定到任何的本地 IPv4 地址，它就只能接受 IPv4 数据。
any-ipv4-address	空元素表示使用这个接口的套接字应该绑定到 IPv4 通配符地址 (0.0.0.0)。
any-ipv6-address	空元素表示使用这个接口的套接字应该绑定到 IPv6 通配符地址 (::)。
inet-address	请输入 IPv6 格式的 IP 地址或者用小数点隔开的 IPv4 地址，或者可以解析为 IP 地址的主机名。
link-local-address	空元素表示接口的部分选择标准应该是或不是和 link-local 关联的地址。
loopback	空元素表示接口的部分选择标准应该是或不是 loopback 接口。
loopback-address	可能实际上不会在主机的 loopback 接口上配置的 loopback 地址。和 <code>inet-addressType</code> 不同的是，即使没有找到和 IP 地址相关的 NIC，给定的值也将被使用。
multicast	空元素表示接口的部分选择标准应该支持或不支持多点传送。
nic	网络接口的名称（如 <code>eth0</code> , <code>eth1</code> , <code>lo</code> ）。
nic-match	常规表达式，表示主机上可以映射可接受的接口的网络接口的名称。
not	地址排斥类型的空元素，用于约束选择标准。
point-to-point	空元素表示接口的部分选择标准是是否为 point-to-point 接口。
public-address	空元素表示接口的部分选择标准应该有或没有公共路由的地址。
site-local-address	空元素表示接口的部分选择标准应该是或不是和 site-local 关联的地址。
subnet-match	网络 IP 地址和地址的网络前缀的位数，以斜杠和数字表示；如 "192.168.0.0/16"。
up	空元素表示接口的部分选择标准应该是或不是正在运行。

接口元素	描述
virtual	空元素表示接口的部分选择标准应该是或不是虚拟接口。

- **配置接口属性**

您可以用 Tab Completion 来完成输入的命令，并开放可用的属性。

- **用管理 CLI 配置接口属性**

使用管理 CLI 来添加新的接口并编写新的接口属性的值。

- a. **添加新的接口**

使用 **add** 操作来创建新的接口。您可以在管理 CLI 会话的根目录里运行 **add** 命令，下面的例子创建了一个名为 *interfacename* 的接口，它将 **inet-address** 声明为 *12.0.0.2*。

```
/interface=interfacename/:add(inet-address=12.0.0.2)
```

- b. **编辑接口属性**

write 操作将新的值写入属性。下面的例子将 **inet-address** 的值更新为 *12.0.0.8*。

```
/interface=interfacename/:write-attribute(name=inet-address,
value=12.0.0.8)
```

- c. **检验接口属性**

通过 **include-runtime=true** 参数运行 **read-resource** 操作来确认值已修改，从而开放服务器模型里所有当前的值。例如：

```
[standalone@localhost:9999 interface=public] :read-
resource(include-runtime=true)
```

- **用管理控制台配置接口属性**

- a. **登录到管理控制台。**

登录到受管域或独立服务器实例的管理控制台。

- b. **进入 Interfaces 屏幕**

- i. **进入 Configuration 标签页。**

从屏幕顶部选择 **Configuration** 标签页。

- ii. **仅用于域模式**

从屏幕左上角的 **Profile** 下拉菜单里选择要修改的配置集。

- c. **从导航菜单里选择 Interfaces。**

展开 **General Configuration** 菜单。从导航菜单里选择 **Interfaces** 菜单条目。

- d. **添加新的接口**

- i. **点添加。**

ii. 输入 **Name**、**Inet Address** 和 **Address Wildcard** 的值。

iii. 点击 **Save**。

e. 编辑接口属性

- i. 从 **Available Interfaces** 列表里选择要编辑的接口并点击 **Edit**。
- ii. 输入 **Name**、**Inet Address** 和 **Address Wildcard** 的值。
- iii. 点击 **Save**。

[提交 bug 报告](#)

5.2. 套接字绑定组

5.2.1. 关于套接字绑定组

套接字绑定和绑定组允许您定义网络接口及它们和 JBoss EAP 6 配置要求的网络接口的关系。

套接字绑定是用于套接字的命名配置。这些命名配置的声明可以在 **domain.xml** 和 **standalone.xml** 配置文件里找到。配置的其他部分可以通过逻辑名引用这些套接字，而无需包含套接字配置的完整细节。这允许您引用不同主机上可能不同的相对套接字配置。

套接字绑定是通过套接字组来收集的。套接字绑定组是按照逻辑名称分组的套接字绑定声明的集合。然后这个命名组可以在整个配置里进行引用。独立服务器配置只包含一个这样的组，而受管域实例则可以包含多个组。您可以在受管域里为每个服务器组创建一个套接字绑定组，或者在多个服务器组间分享套接字绑定组。

在配置受管域的服务器组时，命名组允许对特定的套接字绑定组使用简化的引用。另外一个通常的用途是对同一个系统上的多个独立服务器实例的配置和管理。下面的例子分别展示了独立和域实例的配置文件里的默认套接字绑定组。

例 5.6. 独立配置的默认套接字绑定

standalone.xml 配置文件里的默认套接字绑定组是按照 **standard-sockets** 分组的。**public** 接口也通过相同的逻辑引用方法引用了这个组。

```
<socket-binding-group name="standard-sockets" default-
interface="public">
  <socket-binding name="http" port="8080"/>
  <socket-binding name="https" port="8443"/>
  <socket-binding name="jacorb" port="3528"/>
  <socket-binding name="jacorb-ssl" port="3529"/>
  <socket-binding name="jmx-connector-registry" port="1090"
interface="management"/>
  <socket-binding name="jmx-connector-server" port="1091"
interface="management"/>
  <socket-binding name="jndi" port="1099"/>
  <socket-binding name="messaging" port="5445"/>
  <socket-binding name="messaging-throughput" port="5455"/>
  <socket-binding name="osgi-http" port="8090"
interface="management"/>
  <socket-binding name="remoting" port="4447"/>
  <socket-binding name="txn-recovery-environment" port="4712"/>
  <socket-binding name="txn-status-manager" port="4713"/>
</socket-binding-group>
```

例 5.7. 域配置的默认套接字绑定

domain.xml 配置文件里的默认套接字绑定组包含了四个组：**standard-sockets**、**ha-sockets**、**full-sockets** 和 **full-ha-sockets**。这些组也被名为 **public** 的接口所引用。

```
<socket-binding-groups>
  <socket-binding-group name="standard-sockets" default-
interface="public">
    <!-- Needed for server groups using the 'default' profile -->
    <socket-binding name="ajp" port="8009"/>
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="8443"/>
    <socket-binding name="osgi-http" interface="management"
port="8090"/>
    <socket-binding name="remoting" port="4447"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
      <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
  </socket-binding-group>
  <socket-binding-group name="ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'ha' profile -->
    <socket-binding name="ajp" port="8009"/>
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="8443"/>
    <socket-binding name="jgroups-mping" port="0" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-
port="45700"/>
    <socket-binding name="jgroups-tcp" port="7600"/>
    <socket-binding name="jgroups-tcp-fd" port="57600"/>
    <socket-binding name="jgroups-udp" port="55200" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-
port="45688"/>
    <socket-binding name="jgroups-udp-fd" port="54200"/>
    <socket-binding name="modcluster" port="0" multicast-
address="224.0.1.105" multicast-port="23364"/>
    <socket-binding name="osgi-http" interface="management"
port="8090"/>
    <socket-binding name="remoting" port="4447"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
      <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
  </socket-binding-group>
  <socket-binding-group name="full-sockets" default-
interface="public">
    <!-- Needed for server groups using the 'full' profile -->
    <socket-binding name="ajp" port="8009"/>
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="8443"/>
    <socket-binding name="jacob" interface="unsecure" port="3528"/>
    <socket-binding name="jacob-ssl" interface="unsecure">
```

```

port="3529"/>
    <socket-binding name="messaging" port="5445"/>
    <socket-binding name="messaging-group" port="0" multicast-
address="${jboss.messaging.group.address:231.7.7.7}" multicast-
port="${jboss.messaging.group.port:9876}"/>
    <socket-binding name="messaging-throughput" port="5455"/>
    <socket-binding name="osgi-http" interface="management"
port="8090"/>
    <socket-binding name="remoting" port="4447"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
        <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
<socket-binding-group name="full-ha-sockets" default-
interface="public">
    <!-- Needed for server groups using the 'full-ha' profile -->
    <socket-binding name="ajp" port="8009"/>
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="8443"/>
    <socket-binding name="jacob" interface="unsecure" port="3528"/>
    <socket-binding name="jacob-ssl" interface="unsecure"
port="3529"/>
    <socket-binding name="jgroups-mping" port="0" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-
port="45700"/>
    <socket-binding name="jgroups-tcp" port="7600"/>
    <socket-binding name="jgroups-tcp-fd" port="57600"/>
    <socket-binding name="jgroups-udp" port="55200" multicast-
address="${jboss.default.multicast.address:230.0.0.4}" multicast-
port="45688"/>
    <socket-binding name="jgroups-udp-fd" port="54200"/>
    <socket-binding name="messaging" port="5445"/>
    <socket-binding name="messaging-group" port="0" multicast-
address="${jboss.messaging.group.address:231.7.7.7}" multicast-
port="${jboss.messaging.group.port:9876}"/>
    <socket-binding name="messaging-throughput" port="5455"/>
    <socket-binding name="modcluster" port="0" multicast-
address="224.0.1.105" multicast-port="23364"/>
    <socket-binding name="osgi-http" interface="management"
port="8090"/>
    <socket-binding name="remoting" port="4447"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
        <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
</socket-binding-groups>

```

套接字绑定实例可以在应用服务器目录下的 **standalone.xml** 和 **domain.xml** 文件里创建和编辑。我们推荐的管理绑定的方法是使用管理控制台或管理 CLI。使用管理控制台的优势包括图形化的用户界面，在 **General Configuration** 部分里有专门的 **Socket Binding Group** 屏幕。管理 CLI 提供 API 和基于

命令行的批处理工作流程，并可以对应用服务器配置的更高和更低级别使用脚本。两种界面都可以持久化修改或者保存修改到服务器配置文件里。

[提交 bug 报告](#)

5.2.2. 配置套接字绑定

套接字绑定可以在唯一的套接字绑定组里定义。独立服务器包含一个 **standard-sockets** 组，且无法创建更多的组。相反，您可以创建替代的独立服务器配置文件。对于受管域，您可以创建多个套接字绑定组并按需要配置它们包含的套接字绑定。下表展示了每个套接字绑定的可用属性。

表 5.2. 套接字绑定属性

属性	描述	角色
name	应该用在配置里其他位置的套接字配置的逻辑名。	必需
port	基于这个配置的套接字应该绑定的基础端口。请注意，您可以通过应用于所有端口的增量或减量来配置服务器以覆盖这个基础值。	必需
interface	基于这个配置的套接字应该绑定的接口的逻辑名。如果没有定义，将会使用附带的套接字绑定组里 default-interface 属性的值。	可选
multicast-address	如果套接字用于多点传送，要使用的多点传送地址。	可选
multicast-port	如果套接字用于多点传送，要使用的多点传送端口。	可选
fixed-port	如果为 true ，表示 true 的值必须总是用于套接字且不应该通过增减值来进行覆盖。	可选

- **配置套接字绑定组里的套接字绑定**
请选择管理 CLI 或管理控制台来按需要配置套接字绑定。
 - **使用管理 CLI 配置套接字绑定**
使用管理 CLI 配置套接字绑定。
 - a. **添加新的套接字绑定**
如果有需要，请使用 **add** 操作来创建新的地址设置。您可以在管理 CLI 会话的根目录里运行这个命令，下面的例子创建了一个名为 *newsocket* 的套接字绑定，它的 **port** 属性声明为 *1234*。这些例子适用于独立服务器和受管域的 **standard-sockets** 套接字绑定组。


```
[domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:add(port=1234)
```

b. 编辑 Pattern 属性

使用 **write-attribute** 操作来编写新的属性的值。您可以使用 **tab completion** 来帮助输入并提示所有可用的值。下面的例子将 **port** 的值更新为 **2020**。

```
[domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:write-attribute(name=port,value=2020)
```

c. 确认 Pattern 属性

通过 **include-runtime=true** 参数运行 **read-resource** 操作来确认值已修改以开放服务器模型里所有当前的值。

```
[domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:read-resource
```

o. 使用管理控制台配置套接字绑定

使用管理控制台配置套接字绑定。

a. 登录到管理控制台。

登录到受管域或独立服务器的管理控制台。

b. 进入 Configuration 标签页。

从屏幕顶部选择 **Configuration** 标签页。

c. 从导航菜单里选择 Socket Binding 菜单条目。

展开 **General Configuration** 菜单。选择 **Socket Binding**。如果您在使用受管域，请在 **Socket Binding Groups** 列表里选择所需的组。

d. 添加新的套接字绑定

i. 点击 **Add** 按钮。

ii. 输入 **Name**, **Port** 和 **Binding Group** 的值。

iii. 点击 **Save** 完成。

e. 编辑套接字绑定

i. 从列表里选择要编辑的套接字绑定并点击 **Edit** 按钮。

ii. 输入 **Name**, **Interface** 或 **Port** 的值。

iii. 点击 **Save** 完成。

[提交 bug 报告](#)

5.2.3. JBoss EAP 6 使用的网络端口

JBoss EAP 6 的默认配置使用的端口取决于下列因素：

- 你的服务器组是否使用了默认的套接字绑定组，或者自定义的套接字绑定组。

- 单独部署的要求。



注意

你可以配置一个数字的端口偏移量，以减缓当在同一个服务服务器上运行多个服务器时端口冲突。如果你的服务器使用了数字的端口偏移量，在它的服务器组的套接字绑定组的默认端口上添加偏移量。例如，如果套接字绑定组的 HTTP 端口是 **8080**，而你的服务器使用了端口偏移量为 **100**，那么它的 HTTP 端口是 **8180**。

除非额外指定，这个端口将使用 TCP 协议。

默认的套接字绑定组

- **full-ha-sockets**
- **full-sockets**
- **ha-sockets**
- **standard-sockets**

表 5.3. 默认的套接字绑定组的引用

名称	端口	多点传送 端口	描述	full-ha- sockets	full- sockets	ha- socket	standar d- socket
ajp	8009		Apache JServ 协议，用于 HTTP 群集和负载平衡。	支持	支持	支持	支持
http	8080		用于已部署应用程序的默认端口。	支持	支持	支持	支持
https	8443		已部署的应用程序和客户间的用 SSL 加密的连接。	支持	支持	支持	支持
jacorb	3528		用于 JTS 事务的 CORBA 服务和其他依赖于 ORB 的服务。	支持	支持	不支持	不支持
jacorb-ssl	3529		SSL 加密的 CORBA 服务。	支持	支持	不支持	不支持
jgroups-diagnostics		7500	多点传送。用于 HA 群集里的 Peer 发现。不能使用管理界面进行配置。	支持	不支持	支持	不支持

名称	端口	多点传送 端口	描述	full-ha- sockets	full- sockets	ha- socket	standar d- socket
jgroup s- mping		45700	多点传送。用于在 HA 群集里发现初始成员资格。	支持	不支持	支持	不支持
jgroup s-tcp	7600		HA 群集里使用 TCP 的多点传送 Peer 发现。	支持	不支持	支持	不支持
jgroup s-tcp- fd	57600		用于 TCP 上的 HA 失败检测。	支持	不支持	支持	不支持
jgroup s-udp	55200	45688	HA 群集里使用 UDP 的多点传送 Peer 发现。	支持	不支持	支持	不支持
jgroup s-udp- fd	54200		用于 UDP 上的 HA 失败检测。	支持	不支持	支持	不支持
messaging	5445		JMS 服务。	支持	支持	不支持	不支持
messaging- group			被 HornetQ JMS 广播和发现组引用。	支持	支持	不支持	不支持
messaging- throughput	5455		JMS remoting 所使用的。	支持	支持	不支持	不支持
mod_cluster		23364	用于 JBoss EAP 6 和 HTTP 加载平衡器之间通讯的多点传送端口。	支持	不支持	支持	不支持
osgi- http	8090		由使用 OSGi 子系统的内部组件使用。不能通过管理界面进行配置。	支持	支持	支持	支持
remoting	4447		用于远程 EJB 调用。	支持	支持	支持	支持

名称	端口	多点传送 端口	描述	full-ha- sockets	full- sockets	ha- socket	standar d- socket
txn- recove ry- enviro nment	4712		JTA 事务恢复管理 者。	支持	支持	支持	支持
txn- status - manage r	4713		JTA / JTS 事务管理 者。	支持	支持	支持	支持

管理端口

除了套接字绑定组，每个主机控制台都打开另外两个端口用于管理：

- **9990** - Web 管理控制台的端口
- **9999** - 管理控制台和 API 使用的端口

此外，如果管理控制台启用了 HTTPS，那么 **9443** 将作为默认端口打开。

[提交 bug 报告](#)

5.2.4. 关于套接字绑定组的端口偏移

端口偏移是添加到服务器的套接字组给定的端口值的数字偏移量。这允许单个服务器继承服务器组的套接字绑定，并用偏移量来确保它和组里的其他服务器不冲突。例如，如果套接字绑定组的 HTTP 端口是 8080，而你的服务器使用了端口偏移量为 100，那么它的 HTTP 端口是 8180。

[提交 bug 报告](#)

5.2.5. 配置端口偏移

- **配置端口偏移**

选择管理 CLI 或管理控制台来配置您的端口偏移。

- **使用管理 CLI 配置端口偏移**

使用管理 CLI 来配置端口偏移。

- a. **编辑端口偏移**

使用 **write-attribute** 操作来为端口偏移属性编写新的值。下面的例子更新了 **server-two** 的 **socket-binding-port-offset** 值为 250。这个服务器是默认本地主机组的成员。为使改动生效，服务器需要重启。

```
[domain@localhost:9999 /] /host=master/server-config=server-  
two/:write-attribute(name=socket-binding-port-  
offset,value=250)
```

b. 确认端口偏移属性

通过 `include-runtime=true` 参数运行 `read-resource` 操作来确认值已修改以开放服务器模型里所有当前的值。

```
[domain@localhost:9999 /] /host=master/server-config=server-two/:read-resource(include-runtime=true)
```

o. 使用管理控制台来配置端口偏移

使用管理控制台来配置端口偏移。

a. 登录到管理控制台。

登录到您的受管域的管理控制台。

b. 选择 Domain 标签页

从屏幕顶部选择 **Domain** 标签页。

c. 编辑端口偏移属性

i. 在 **Available Server Configurations** 列表里选择服务器并点击属性列表顶部的 **Edit** 按钮。

ii. 在 **Port Offset** 字段里输入想要的值。

iii. 点击 **Save** 完成。

[提交 bug 报告](#)

5.2.6. 配置 Remoting 里的消息大小

Remoting 子系统提供了限制用于远程协议的消息的大小的选项。您可以设置最大的转入消息

(`MAX_INBOUND_MESSAGE_SIZE`) 及最大的转出消息 (`MAX_OUTBOUND_MESSAGE_SIZE`) 来确保接收和发送的消息具有合适的大小。

配置远程协议里的消息大小有助于有效地利用系统内存并防止在执行重要任务时内存溢出。

如果发送者发送的消息超过了最大限制 (`MAX_INBOUND_MESSAGE_SIZE`)，服务器将抛出异常并取消数据的传输。然而连接将保持打开且如果需要的话发送者可以选择关闭消息。

如果接收的消息超过了最大限制 (`MAX_INBOUND_MESSAGE_SIZE`)，消息将被异步关闭而连接仍将保持打开。

[提交 bug 报告](#)

5.3. IPV6

5.3.1. 配置 IPv6 网络的 JVM Stack 首选项

概述

本节涵盖为 JBoss EAP 6 安装启用 IPv6 网络。

过程 5.1. 禁用 IPv4 Stack Java 属性

1. 打开相关的安装文件：

- 对于独立服务器：
打开 **EAP_HOME/bin/standalone.conf**。
- 对于受管域：
打开 **EAP_HOME/bin/domain.conf**。

2. 修改 IPv4 Stack Java 属性为 false:

```
-Djava.net.preferIPv4Stack=false
```

例如：

```
# Specify options to pass to the Java VM.
#
if [ "x$JAVA_OPTS" = "x" ]; then
    JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -
Djava.net.preferIPv4Stack=false
-Dorg.jboss.resolver.warning=true -
Dsun.rmi.dgc.client.gcInterval=3600000
-Dsun.rmi.dgc.server.gcInterval=3600000 -
Djava.net.preferIPv6Addresses=true"
fi
```

[提交 bug 报告](#)

5.3.2. 配置 IPv6 网络的接口声明

概述

遵循下列步骤来配置 IPv6 的接口 inet 地址：

必须具备的条件

- [第 2.1.1 节“启动 JBoss EAP 6”](#)
- [第 3.4.2 节“登录到管理控制台”](#)

过程 5.2. 配置 IPv6 网络的接口

1. 从屏幕顶部选择 **Configuration** 标签页。
2. 展开 **General Configuration** 菜单并选择 **Interfaces**。
3. 从 **Available Interfaces** 列表里选择接口。
4. 点击细节列表里的 **Edit** 按钮。
5. 设置 inet 地址为：

```
${jboss.bind.address.management:[ADDRESS]}
```

6. 点击 **Save** 完成。
7. 重启服务器来应用这些修改。

[提交 bug 报告](#)

5.3.3. 配置 IPv6 地址的 JVM Stack 首选项

概述

本节涵盖通过配置文件配置 JBoss EAP 6 安装首选 IPv6 地址。

过程 5.3. 配置 JBoss EAP 6 安装首选 IPv6 地址

1. 打开相关的安装文件：
 - 对于独立服务器：
打开 `EAP_HOME/bin/standalone.conf`。
 - 对于受管域：
打开 `EAP_HOME/bin/domain.conf`。
2. 附加下列 Java 属性到 Java VM 选项：

```
-Djava.net.preferIPv6Addresses=true
```

例如：

```
# Specify options to pass to the Java VM.
#
if [ "x$JAVA_OPTS" = "x" ]; then
    JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -
Djava.net.preferIPv4Stack=false
-Dorg.jboss.resolver.warning=true -
Dsun.rmi.dgc.client.gcInterval=3600000
-Dsun.rmi.dgc.server.gcInterval=3600000 -
Djava.net.preferIPv6Addresses=true"
fi
```

[提交 bug 报告](#)

第 6 章 数据源管理

6.1. 介绍

6.1.1. 关于 JDBC

JDBC API 是定义 Java 应用程序如何访问数据库的标准。应用程序配置引用 JDBC 驱动的数据源。然后可以再次针对驱动而不是数据库编写应用程序代码。驱动将代码转换为数据库语言。这表示如果安装了正确的驱动，应用程序就可以使用受支持的数据库了。

JDBC 4.0 规格是在这里定义的：<http://jcp.org/en/jsr/detail?id=221>。

要开始使用 JDBC 和数据源，请参考 JBoss EAP 6 的《管理和配置指南》里的《JDBC 驱动》章节。

[提交 bug 报告](#)

6.1.2. JBoss EAP 6 支持的数据库

关于 JBoss EAP 6 支持的兼容 JDBC 的数据库列表，请参考：<https://access.redhat.com/site/articles/111663>。

[提交 bug 报告](#)

6.1.3. 数据源的类型

两种常用的资源类型是非 XA 数据源和XA 数据源。

非 XA 数据源用于不使用事务的应用程序，或者以单个数据库使用事务的应用程序。

XA 数据源用于事务分布在多个数据库的应用程序。XA 数据源会导致额外的负荷。

当你在管理控制台或管理 CLI 里创建数据源时，你可以指定它的类型。

[提交 bug 报告](#)

6.1.4. 数据源示例

JBoss EAP 6 里包含了一个 H2 数据源，它是一个轻量级的关系型数据库管理系统，它为开发者提供了快速构建应用程序的能力，而且是平台的示例数据源。



警告

然而，JBoss EAP 附带的示例数据源不应该用于产品环境。它是一个非常小、自包容的数据源，它支持所有测试和构建应用程序所需的标准，但它并不健壮也不具有足够的可扩充性以用于产品环境。

关于被支持和认证的数据源，请参考 第 6.1.2 节“JBoss EAP 6 支持的数据库”。

[提交 bug 报告](#)

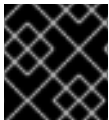
6.1.5. -ds.xml 文件的部署

在 JBoss EAP 6 里，数据源被定义为服务器子系统的资源。在以前的版本里，服务器配置的 `deploy` 目录里要求有 `*-ds.xml` 数据源配置文件。`*-ds.xml` 文件仍可以按照 *Schemas* 这里 <http://www.ironjacamar.org/documentation.html> 的 1.1 数据源模式部署在 JBoss EAP 6 里。



警告

这个功能应该只用于部署。我们不推荐将其用于产品环境，因为 JBoss 管理工具并不支持它。



重要

当部署 `*-ds.xml` 文件时，使用对已部署 / 定义的 `<driver>` 条目的引用是强制的。

[提交 bug 报告](#)

6.2. JDBC 驱动

6.2.1. 用管理控制台安装 JDBC 驱动

介绍

在你的应用程序可以连接 JDBC 数据源之前，你的数据源供应商的 JDBC 驱动需要安装在 JBoss EAP 可以使用的位置上。JBoss EAP 6 允许你象其他部署一样部署这些驱动。这意味着如果你使用了受管域，你可以将它们部署在服务器组里的多个服务器上。

预备条件

在执行这个任务之前，你需要满足以下预备条件：

- 从数据库供应商下载 JDBC 驱动。



注意

任何兼容 JDBC 4 的驱动都自动会被承认且根据名称和版本安装至系统里。JDBC JAR 通过 Java 服务供应商机制来识别。这样的 JAR 里有 `META-INF/services/java.sql.Driver` 文本，它包含该 JAR 里驱动类的名称。

过程 6.1. 修改 JDBC 驱动 JAR

如果 JDBC 驱动 JAR 不兼容 JDBC 4，您可用下列方法使其成为可部署的。

1. 修改或创建空的临时目录。
2. 创建一个 `META-INF` 子目录。
3. 创建一个 `META-INF/services` 子目录。
4. 创建一个 `META-INF/services/java.sql.Driver` 文件，它包含一行指明 JDBC 驱动的全限定类名的

内容。

5. 使用 JAR 命令行工具来更新 JAR：

```
jar \-uf jdbc-driver.jar META-INF/services/java.sql.Driver
```

过程 6.2. 部署 JDBC 驱动

1. 访问管理控制台。

[第 3.4.2 节 “登录到管理控制台”](#)

2. 将 JAR 文件部署到服务器或服务组。

如果你使用了受管域，你可以将 JAR 文件部署到服务器组。否则，部署到自己的服务器。请参考 [第 10.2.2 节 “用管理控制台启用已部署的应用程序”](#)。

结果：

JDBC 驱动被部署，可由你的应用程序所使用。

[提交 bug 报告](#)

6.2.2. 将 JDBC 驱动安装为核心模块

预备条件

在执行这个任务之前，你需要满足以下预备条件：

- 从数据库供应商下载 JDBC 驱动。JDBC 驱动的下载位置是：[第 6.2.3 节 “JDBC 驱动的下载位置”](#)。
- 解压归档文件。

过程 6.3. 将 JDBC 驱动安装为核心模块

1. 在 **EAP_HOME/modules/** 目录下创建一个文件路径结构。例如，对于 MySQL JDBC 驱动，创建下列目录结构：**EAP_HOME/modules/com/mysql/main/**。
2. 将 JDBC 驱动的 JAR 文件复制到 **main/** 子目录。
3. 在 **main/** 子目录里，创建一个类似于下列示例的 **module.xml** 文件：[第 7.1.1 节 “模块”](#)。module XSD 在 **EAP_HOME/docs/schema/module-1_2.xsd** 文件里进行定义。
4. 启动服务器。
5. 启动管理 CLI。
6. 运行 CLI 命令将 JDBC 驱动模块添加到服务器配置里。

你选择的命令取决于 JDBC 驱动 JAR 里的 **/META-INF/services/java.sql.Driver** 文件里列出的类的数量。例如，MySQL 5.1.20 JDBC JAR 里的 **/META-INF/services/java.sql.Driver** 文件列出了两个类：

- **com.mysql.jdbc.Driver**
- **com.mysql.fabric.jdbc.FabricMySQLDriver**

当有多个条目时，您必须也指定驱动类的名称。没这样做会导致这样的错误：

```
JBAS014749: Operation handler failed: Service jboss.jdbc-driver.mysql is already registered
```

- 为包含一个驱动类条目的 JDBC JAR 运行 CLI 命令。

```
/subsystem=datasources/jdbc-driver=DRIVER_NAME:add(driver-name=DRIVER_NAME, driver-module-name=MODULE_NAME, driver-xa-datasource-class-name=XA_DATASOURCE_CLASS_NAME)
```

例 6.1. 用于具有一个驱动类的 JDBC JAR 的独立模式的 CLI 命令示例

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql, driver-module-name=com.mysql, driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource)
```

例 6.2. 用于具有一个驱动类的 JDBC JAR 的域模式的 CLI 命令示例

```
/profile=ha/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql, driver-module-name=com.mysql, driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource)
```

- 为具有多个驱动类条目的 JDBC JAR 运行 CLI 命令。

```
/subsystem=datasources/jdbc-driver=DRIVER_NAME:add(driver-name=DRIVER_NAME, driver-module-name=MODULE_NAME, driver-xa-datasource-class-name=XA_DATASOURCE_CLASS_NAME, driver-class-name=DRIVER_CLASS_NAME)
```

例 6.3. 用于具有多个驱动类条目的 JDBC JAR 的独立模式的 CLI 命令示例

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql, driver-module-name=com.mysql, driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource, driver-class-name=com.mysql.jdbc.Driver)
```

例 6.4. 用于具有多个驱动类条目的 JDBC JAR 的域模式的 CLI 命令示例

```
/profile=ha/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql, driver-module-name=com.mysql, driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource, driver-class-name=com.mysql.jdbc.Driver)
```

结果

已安装好 JDBC 驱动并设置为核心模块，且可以被应用程序数据源引用。

[提交 bug 报告](#)

6.2.3. JDBC 驱动的下载位置

下表列出了 JBoss EAP 6 常用的数据库的 JDBC 驱动的下载位置。这些链接指向不受 Red Hat 监控或控制的第三方网站。关于您的数据库的最新驱动，请参考相关数据库供应商的文档和网站。

表 6.1. JDBC 驱动的下载位置

供应商	下载位置
MySQL	http://www.mysql.com/products/connector/
PostgreSQL	http://jdbc.postgresql.org/
Oracle	http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html
IBM	http://www-306.ibm.com/software/data/db2/java/
Sybase	http://www.sybase.com/products/allproductsa-z/softwaredeveloperkit/jconnect
Microsoft	http://msdn.microsoft.com/data/jdbc/

[提交 bug 报告](#)

6.2.4. 访问供应商专有的类

介绍

本节涵盖使用 JDBC 专有类所需的步骤。当应用程序需要使用非 JDBC API 一部分的供应商专有功能时这是有必要的。



警告

这是高级的应用。只有需要 JDBC API 里找不到的功能时才应该实现这个过程。



重要

在使用重验证机制并访问供应商专有的类时这个过程是必需的。



重要

因为这个连接是被 Iron Jacamar 容器控制的，请严格遵循供应商专有的 API 准则。

前提条件

- 第 6.2.2 节 “将 JDBC 驱动安装为核心模块”。

过程 6.4. 在应用程序里添加依赖关系

- ○ 配置 MANIFEST.MF 文件
 - a. 在文本编辑器里打开应用程序的 META-INF/MANIFEST.MF 文件。
 - b. 为 JDBC 模块添加一个依赖关系并保存文件。

依赖关系：MODULE_NAME

例 6.5. 依赖关系示例

依赖关系：com.mysql

- ○ a. 创建一个 jboss-deployment-structure.xml 文件
在应用程序的 META-INF/ or WEB-INF 文件夹里创建一个名为 jboss-deployment-structure.xml 的文件。

例 6.6. jboss-deployment-structure.xml 文件示例

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="com.mysql" />
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

例 6.7. 访问供应商专有的 API

下面的例子访问了 MySQL API。

```
import java.sql.Connection;
import org.jboss.jca.adapters.jdbc.WrappedConnection;

Connection c = ds.getConnection();
WrappedConnection wc = (WrappedConnection)c;
com.mysql.jdbc.Connection mc = wc.getUnderlyingConnection();
```

[提交 bug 报告](#)

6.3. NON-XA 数据源

6.3.1. 用管理界面创建一个 Non-XA 数据源

概述

本节涵盖用管理控制台或管理 CLI 创建 Non-XA 数据源所需的步骤。

前提条件

- JBoss EAP 6 服务器必须正在运行。



注意

在 Oracle 数据源 10.2 之前的版本里，`<no-tx-separate-pools/>` 参数是必需的，因为非事务性和事务性连接的混合可能导致错误。对于某些应用程序来说，这个参数已不再是必需的了。

过程 6.5. 用管理 CLI 或管理控制台创建一个数据源

- ○ 管理 CLI

- a. 启动 CLI 工具并连接到您的服务器。
- b. 运行下列命令来创建 Non-XA 数据源，配置合适的变量：

```
data-source add --name=DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_NAME --connection-url=CONNECTION_URL
```

- c. 启用数据源：

```
data-source enable --name=DATASOURCE_NAME
```

- 管理控制台

- a. 登陆到管理控制台。
- b. 进入管理控制台的 **Datasources** 面板
 - i. 从控制台顶部选择 **Configuration** 标签页。
 - ii. 对于域模式，从左上角的下拉菜单里选择合适的配置集。
 - iii. 展开控制台左侧的 **Subsystems** 菜单，然后展开 **Connector** 菜单。
 - iv. 从控制台左侧的菜单里选择 **Datasources**。
- c. 创建新的数据源
 - i. 点击 **Datasources** 面板顶部的 **Add** 按钮。
 - ii. 在 **Create Datasource** 向导里输入新的数据源属性并点击 **Next** 按钮。
 - iii. 在 **Create Datasource** 向导里输入 JDBC 驱动细节并点击 **Next** 按钮。
 - iv. 在 **Create Datasource** 向导里输入连接设置。
 - v. 点击 **Test Connection** 按钮测试到数据源的连接并检验设置是否正确。
 - vi. 点击 **Done** 完成。

结果

Non-XA 数据源已被添加至服务器。它在 `standalone.xml` 或 `domain.xml` 文件以及管理界面里都可见。

[提交 bug 报告](#)

6.3.2. 用管理界面修改 Non-XA 数据源

概述

本节涵盖用管理控制台或管理 CLI 修改 Non-XA 数据源所需的步骤。

必须具备的条件

- [第 2.1.1 节 “启动 JBoss EAP 6”](#)。



注意

Non-XA 数据源可以和 JTA 事务继承。要继承数据源和 JTA，请确保 `jta` 参数被设置为 `true`。

过程 6.6. 修改 Non-XA 数据源

- - 管理 CLI
 - [第 3.5.2 节 “启动管理 CLI”](#)。
 - 使用 `write-attribute` 命令来配置数据源属性：

```
/subsystem=datasources/data-source=DATASOURCE_NAME:write-attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

- 重载服务器来确认修改：

```
:reload
```

- 管理控制台

- [第 3.4.2 节 “登录到管理控制台”](#)。
- 进入管理控制台的 **Datasources** 面板
 - 从控制台顶部选择 **Configuration** 标签页。
 - 对于域模式，从左上角的下拉菜单里选择合适的配置集。
 - 展开控制台左侧的 **Subsystems** 菜单，然后展开 **Connector** 菜单。
 - 从展开的菜单选择 **Datasources**。

- 编辑数据源

- 从 **Available Datasources** 列表里选择相关的数据源。下面显示了数据源属性。
- 点击 **Edit** 按钮来编辑数据源属性。

- iii. 点击**Save**完成。

结果

已完成对 **Non-XA** 数据源的配置。这些修改在 **standalone.xml** 或 **domain.xml** 文件以及管理界面里都可见。

- 要创建新的数据源，请参考：[第 6.3.1 节 “用管理界面创建一个 Non-XA 数据源”](#)。
- 要删除数据源，请参考：[第 6.3.3 节 “用管理界面删除 Non-XA 数据源”](#)。

[提交 bug 报告](#)

6.3.3. 用管理界面删除 Non-XA 数据源

概述

本节涵盖用管理控制台或管理 CLI 从 JBoss EAP 6 删除 **Non-XA** 数据源所需的步骤。

必须具备的条件

- [第 2.1.1 节 “启动 JBoss EAP 6”](#)。

过程 6.7. 删除 Non-XA 数据源

- ○ 管理 CLI
 - a. [第 3.5.2 节 “启动管理 CLI”](#)。
 - b. 运行下列命令来删除 **Non-XA** 数据源：

```
data-source remove --name=DATASOURCE_NAME
```
- ○ 管理控制台
 - a. [第 3.4.2 节 “登录到管理控制台”](#)。
 - b. 进入管理控制台的 **Datasources** 面板
 - i. 从控制台顶部选择 **Configuration** 标签页。
 - ii. 对于域模式，从左上角的下拉菜单里选择合适的配置集。
 - iii. 展开控制台左侧的 **Subsystems** 菜单，然后展开 **Connector** 菜单。
 - iv. 选择 **Datasources**。
 - c. 选择要删除的数据源，然后点击 **Remove**。

结果

非 **XA** 数据源已从服务器删除。

- 要添加新的数据源，请参考：[第 6.3.1 节 “用管理界面创建一个 Non-XA 数据源”](#)。

[提交 bug 报告](#)

6.4. XA 数据源

6.4.1. 用管理界面创建 XA 数据源

前提条件：

- 第 2.1.1 节 “启动 JBoss EAP 6”

介绍

本节涵盖用管理控制台或管理 CLI 创建 XA 数据源所需的步骤。



注意

在 Oracle 数据源 10.2 之前的版本里，`<no-tx-separate-pools/>` 参数是必需的，因为非事务性和事务性连接的混合可能导致错误。对于某些应用程序来说，这个参数已不再是必需的了。

过程 6.8. 用管理 CLI 或管理控制台创建 XA 数据源

- 管理 CLI

a. 第 3.5.2 节 “启动管理 CLI”。

b. 运行下列命令来创建 XA 数据源，配置合适的变量：

```
xa-data-source add --name=XA_DATASOURCE_NAME --jndi-
name=JNDI_NAME --driver-name=DRIVER_NAME --xa-datasource-
class=XA_DATASOURCE_CLASS
```

c. 配置 XA 数据源属性

i. 设置服务器名称

运行下列命令来配置主机的服务器名称：

```
/subsystem=datasources/xa-data-
source=XA_DATASOURCE_NAME/xa-datasource-
properties=ServerName:add(value=HOSTNAME)
```

ii. 设置数据库名称

运行下列命令来配置数据库名称：

```
/subsystem=datasources/xa-data-
source=XA_DATASOURCE_NAME/xa-datasource-
properties=DatabaseName:add(value=DATABASE_NAME)
```

d. 启用数据源：

```
xa-data-source enable --name=XA_DATASOURCE_NAME
```

- 管理控制台

a. 第 3.4.2 节 “登录到管理控制台”。

b. 进入管理控制台的 Datasources 面板

- i. 从控制台顶部选择 **Configuration** 标签页。
- ii. 对于域模式，从左上角的下拉菜单里选择合适的配置集。
- iii. 展开控制台左侧的 **Subsystems** 菜单，然后展开 **Connector** 菜单。
- iv. 选择 **Datasources**。

c. 选择 XA Datasource 标签页。**d. 创建新的 XA 数据源**

- i. 点添加。
- ii. 在 **Create XA Datasource** 向导里输入新的 XA 数据源属性并点击 **Next** 按钮。
- iii. 在 **Create XA Datasource** 向导里输入 JDBC 驱动细节并点击 **Next** 按钮。
- iv. 输入 XA 属性并点击 **Next**。
- v. 在 **Create XA Datasource** 向导里输入连接设置。
- vi. 点击 **Test Connection** 按钮测试到 XA 数据源的连接并检验设置是否正确。
- vii. 点击 **Done** 完成。

结果

XA 数据源已被添加至服务器。它在 `standalone.xml` 或 `domain.xml` 文件以及管理界面里都可见。

还可查看：

- [第 6.4.2 节“用管理界面修改 XA 数据源”](#)
- [第 6.4.3 节“用管理界面删除 XA 数据源”](#)

[提交 bug 报告](#)**6.4.2. 用管理界面修改 XA 数据源****介绍**

本节涵盖用管理控制台或管理 CLI 修改 XA 数据源所需的步骤。

前提条件

- [第 2.1.1 节“启动 JBoss EAP 6”](#)。

过程 6.9. 用管理 CLI 或管理控制台修改 XA 数据源

- ○ 管理 CLI
 - a. [第 3.5.2 节“启动管理 CLI”](#)。
 - b. 配置 XA 数据源属性

使用 **write-attribute** 命令来配置数据源属性：

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME:write-attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

c. 配置 XA 数据源属性

运行下列命令来配置 XA 数据源子资源：

```
/subsystem=datasources/xa-data-source=DATASOURCE_NAME/xa-datasource-properties=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

d. 重载服务器来确认修改：

```
:reload
```

o. 管理控制台

a. [第 3.4.2 节“登录到管理控制台”](#)。

b. 进入管理控制台的 **Datasources** 面板

- i. 从控制台顶部选择 **Configuration** 标签页。
- ii. 对于域模式，从左上角的下拉菜单里选择合适的配置集。
- iii. 展开控制台左侧的 **Subsystems** 菜单，然后展开 **Connector** 菜单。
- iv. 选择 **Datasources**。

c. 选择 **XA Datasource** 标签页。

d. 编辑数据源

- i. 从 **Available XA Datasources** 列表里选择相关的数据源。XA 数据源属性显示在下面的 **Attributes** 面板上。
- ii. 选择 **Edit** 按钮来编辑数据源属性。
- iii. 编辑 XA 数据源属性并在完成时选择 **Save** 按钮。

结果

已完成对 XA 数据源的配置。这些修改在 **standalone.xml** 或 **domain.xml** 文件以及管理界面里都可见。

- 要创建新的数据源，请参考：[第 6.4.1 节“用管理界面创建 XA 数据源”](#)。
- 要删除数据源，请参考：[第 6.4.3 节“用管理界面删除 XA 数据源”](#)。

[提交 bug 报告](#)

6.4.3. 用管理界面删除 XA 数据源

介绍

本节涵盖用管理控制台或管理 CLI 从 JBoss EAP 6 删除 XA 数据源所需的步骤。

前提条件

- [第 2.1.1 节“启动 JBoss EAP 6”](#)。

过程 6.10. 用管理 CLI 或管理控制台删除 XA 数据源

- ○ 管理 CLI

- a. [第 3.5.2 节“启动管理 CLI”](#)。
- b. 运行下列命令来删除 XA 数据源：

```
xa-data-source remove --name=XA_DATASOURCE_NAME
```

- 管理控制台

- a. [第 3.4.2 节“登录到管理控制台”](#)。
- b. 进入管理控制台的 **Datasources** 面板
 - i. 从控制台顶部选择 **Configuration** 标签页。
 - ii. 对于域模式，从左上角的下拉菜单里选择合适的配置集。
 - iii. 展开控制台左侧的 **Subsystems** 菜单，然后展开 **Connector** 菜单。
 - iv. 选择 **Datasources**。
- c. 选择 **XA Datasource** 标签页。
- d. 选择要删除的 XA 数据源，然后点击 **Remove** 按钮来永久地删除这个 XA 数据源。

结果

XA 数据源已从服务器删除。

- 要添加新的 XA 数据源，请参考：[第 6.4.1 节“用管理界面创建 XA 数据源”](#)。

[提交 bug 报告](#)

6.4.4. XA Recovery

6.4.4.1. 关于 XA Recovery 模块

每个 XA 资源都需要一个 **recovery** 模块与其配置相关联。这个 **recovery** 模块必须继承 `com.arjuna.ats.jta.recovery.XAResourceRecovery`。

JBoss EAP 6 为 JDBC 和 JMS XA 资源提供了 **recovery** 模块。对于这些类型的资源，**recovery** 模块会自动注册。如果您需要使用自定义模块，您可以在自己的数据源里注册它。

[提交 bug 报告](#)

6.4.4.2. 配置 XA Recovery 模块

对于多数 JDBC 和 JMS 资源，**recovery** 模块自动和资源相关联。在这些情况下，您只需要配置选项以允许 **recovery** 模块连接到您的资源来执行恢复。

对于非 JDBC 或 JMS 的自定义资源，请联系 Red Hat 全球支持服务获取受支持的配置的信息。

每个配置属性都可以在数据源创建过程中或之后设置。你可以用基于 web 的管理控制台或命令行管理 CLI 进行设置。关于配置 XA 数据源的信息，请参考第 6.4.1 节“用管理界面创建 XA 数据源”或第 6.4.2 节“用管理界面修改 XA 数据源”。

参考下列表里的常用数据源配置属性，以及和专有数据库供应商相关的配置细节。

表 6.2. 常用的配置属性

属性	描述
recovery-username	recovery 模块应该用来连接资源进行恢复的用户名。
recovery-password	recovery 模块应该用来连接资源进行恢复的密码。
recovery-security-domain	recovery 模块应该用来连接资源进行恢复的安全域。
recovery-plugin-class-name	如果你需要使用自定义的 recovery 模块，请将这个属性设置为模块的全限定名。这个模块应该继承 com.arjuna.ats.jta.recovery.XAResourceRecovery 类。
recovery-plugin-properties	如果你使用了要求设置属性的自定义 recovery 模块，请将这个属性设置为用逗号隔开的 key=value 对的列表。

供应商专有的配置信息

Oracle

如果错误地配置了 Oracle 数据源，您可能会在日志输出里看到这样的错误：

```
WARN [com.arjuna.ats.jta.logging.loggerI18N]
[com.arjuna.ats.internal.jta.recovery.xarecovery1] Local
XARecoveryModule.xaRecovery got XA exception
javax.transaction.xa.XAException, XAException.XAER_RMERR
```

要解决这个错误，请确保 **recovery-username** 里配置的 Oracle 用户可以访问恢复所需的表。下面是安装了 Oracle bug 5945463 补丁的 Oracle 11g 或 Oracle 10g R2 实例的正确 Grant SQL 语句。

```
GRANT SELECT ON sys.dba_pending_transactions TO recovery-username;
GRANT SELECT ON sys.pending_trans$ TO recovery-username;
GRANT SELECT ON sys.dba_2pc_pending TO recovery-username;
GRANT EXECUTE ON sys.dbms_xa TO recovery-username;
```

如果你使用了 11g 以前的 Oracle 11 版本，请将最后的 **EXECUTE** 语句修改为：

```
GRANT EXECUTE ON sys.dbms_system TO recovery-username;
```

PostgreSQL

关于启用 `prepared` (也就是 XA) 事务的说明请阅读 PostgreSQL 文档。PostgreSQL 的 JDBC 驱动的 8.4-701 版本在 `org.postgresql.xa.PGXAConnection` 里有一个程序错误，它在某些情况下会中断恢复。在更新的版本里我们会修复这个问题。

MySQL

根据 <http://bugs.mysql.com/bug.php?id=12161>，XA 事务恢复在 MySQL 5 的某些版本里无法运行。MySQL 6.1 里已解决了这个问题。详情请参考 bug URL 或 MySQL 文档。

IBM DB2

IBM DB2 期望 `XAResource.recover` 方法只是在应用服务器发生崩溃或故障后重启时的重同步阶段才被调用。这是 DB2 实现里的设计问题，超出了本文档的范畴。

Sybase

Sybase 期望在数据库启用 XA 事务。如果没有正确的数据库配置，XA 事务将无法工作。`enable xact coordination` 启用或禁用 Adaptive Server 事务协调 (transaction coordination) 服务。当启用这个参数时，Adaptive Server 确保对远程 Adaptive Server 数据提交或用原始事务进行回滚。要启用事务协调，请使用：

```
sp_configure 'enable xact coordination', 1
```

[提交 bug 报告](#)

6.5. 数据源安全性

6.5.1. 关于数据源安全性

数据源安全性的首选方案是使用安全域或密码阀。下面是它们各自的例子。关于更多的信息，请参考：

- 安全域：第 11.6.1 节“关于安全域”。
- 密码阀：第 11.13.1 节“关于保护明码文件里的敏感字符”。

例 6.8. 安全域示例

```
<security>
  <security-domain>mySecurityDomain</security-domain>
</security>
```

例 6.9. 密码阀示例

```
<security>
  <user-name>admin</user-name>

  <password>${VAULT::ds_ExampleDS::password::N2NhZDYzOTMtNWE0S00ZGQ0LWE4M
mEtMWNlMDMyNDdmNmI2TEl0RV9CukVBS3ZhdWx0}</password>
</security>
```

[提交 bug 报告](#)

6.6. 数据源配置

6.6.1. 数据源参数

表 6.3. XA 和非 XA 数据源共用的数据源参数

参数	描述
jndi-name	数据源的唯一 JNDI 名称。
pool-name	数据源的管理池的名称。
enabled	是否启用数据源
use-java-context	是否绑定数据源到全局 JNDI。
spy	启用 JDBC 层的 spy 功能。这会将所有 JDBC 通讯记录到数据源。请注意 logging 子系统里的日志类别 jboss.jdbc.spy 的级别也必须设置为 DEBUG 。
use-ccm	启用缓存连接管理者。
new-connection-sql	当连接被添加到连接池时会执行的 SQL 语句。
transaction-isolation	下列值之一： <ul style="list-style-type: none">TRANSACTION_READ_UNCOMMITTEDTRANSACTION_READ_COMMITTEDTRANSACTION_REPEATABLE_READTRANSACTION_SERIALIZABLETRANSACTION_NONE
url-delimiter	用于高可用性（HA）群集数据库的 connection-url 的分隔符。
url-selector-strategy-class-name	实现 org.jboss.jca.adapters.jdbc.URLSelectorStrategy 接口的类。
security	包含设置安全性的子元素。请参考 表 6.8 “安全性参数” 。
validation	包含设置有效性的子元素。请参考 表 6.9 “用于检验的参数” 。

参数	描述
timeout	包含设置超时的子元素。请参考 表 6.10 “超时参数” 。
statement	包含设置语句的子元素。请参考 表 6.11 “语句参数” 。

表 6.4. Non-XA 数据源参数

参数	描述
jta	为非 XA 数据源启动 JTA 集成。不适用于 XA 数据源。
connection-url	JDBC 驱动连接的 URL。
driver-class	JDBC 驱动类的全限定名。
connection-property	传递给 Driver.connect(url, props) 方法的任意连接属性。每个 connection-property 都指定一个字符串/值对。属性名称来自元素名称，而值来自元素的内容。
pool	包含设置池的子元素。请参考 表 6.6 “non-XA 和 XA 数据源公用的池参数” 。

表 6.5. XA 数据源参数

参数	描述
xa-datasource-property	分配给 XADataSource 类实现的属性。通过 name=value 指定。如果存在 setName 格式的 setter 方法，这个属性将通过调用 setName(value) 格式的 setter 方法来设置。
xa-datasource-class	javax.sql.XADataSource 实现的全限定名。
driver	对包含 JDBC 驱动的 classloader 模块的唯一引用。有效的格式是 driverName#majorVersion.minorVersion 。
xa-pool	包含设置池的子元素。请参考 表 6.6 “non-XA 和 XA 数据源公用的池参数” 和 表 6.7 “XA 池参数” 。
recovery	包含设置恢复的子元素。请参考 表 6.12 “恢复参数” 。

表 6.6. non-XA 和 XA 数据源公用的池参数

参数	描述
min-pool-size	池里可保留的连接的最小数量。
max-pool-size	池里可保留的连接的最大数量。
prefill	是否预先填充连接池。空的元素表示 true 值。默认为 false 。
use-strict-min	pool-size 是否是严格规定的。默认为 false 。
flush-strategy	在发生错误时是否冲刷池。有效值为： <ul style="list-style-type: none"> • FailingConnectionOnly • IdleConnections • EntirePool 默认值为 FailingConnectionOnly 。
allow-multiple-users	指定是否有多个用户将通过 <code>getConnection(user, password)</code> 访问数据源，且内部池类型是否应该计入在内。

表 6.7. XA 池参数

参数	描述
is-same-rm-override	<code>javax.transaction.xa.XAResource.isSameRM(XAResource)</code> 类是否返回 true 或 false 。
interleaving	是否启用 XA 连接工厂的 interleaving。
no-tx-separate-pools	是否为每个上下文创建单独的子池。对于 Oracle 数据源来说这是必需的，它不允许 XA 连接既在 JTA 内部又在外部使用。 使用这个选项将导致您的池大小两倍于 max-pool-size ，因为这实际会创建两个池。
pad-xid	是否拆分 Xid。
wrap-xa-resource	是否将 XAResource 包裹在 <code>org.jboss.tm.XAResourceWrapper</code> 实例里。

表 6.8. 安全性参数

参数	描述
user-name	创建新连接使用的用户名。
password	创建新连接使用的密码。
security-domain	非 XA 数据源参数
reauth-plugin	定义一个重验证插件以用于重新验证物理连接。

表 6.9. 用于检验的参数

参数	描述
valid-connection-checker	提供 <code>SQLException.isValidConnection(Connection e)</code> 方法来检验连接的 <code>org.jboss.jca.adapters.jdbc.ValidConnectionChecker</code> 接口的实现。异常表示连接已被销毁。它覆盖了 <code>check-valid-connection-sql</code> 参数（如果存在）。
check-valid-connection-sql	检查池连接有效性的 SQL 语句。当从池里获取受管连接进行使用时它会被调用。
validate-on-match	指定当连接工厂试图对给定的集合匹配受管连接时是否执行连接级别的检验。 我们通常不会同时指定 <code>validate-on-match</code> 和 <code>background-validation</code> 为 <code>true</code> 。但客户在使用连接前必须进行检验时则需要 <code>Validate-on-match</code> 。这个参数默认为 <code>false</code> 。
background-validation	指定连接在背景线程上进行检验。背景检验不和 <code>validate-on-match</code> 一起使用是一种性能优化。如果 <code>validate-on-match</code> 为 <code>true</code> 时，使用 <code>background-validation</code> 可能导致冗余的检查。背景检验可能会让用户用到有问题的连接（连接在返回给客户和检验扫描之间可能会出现问题），所以客户应用程序必须考虑到这种可能性。
background-validation-millis	背景检验运行的时间（毫秒）。
use-fast-fail	如果为 <code>true</code> ，在第一次尝试时如果连接无效则失败。默认为 <code>false</code> 。

参数	描述
stale-connection-checker	提供 Boolean <code>isStaleConnection(SQLException e)</code> 方法的 <code>org.jboss.jca.adapters.jdbc.StaleConnectionChecker</code> 实例。如果这个方法返回 <code>true</code> ，异常将包裹在 <code>org.jboss.jca.adapters.jdbc.StaleConnectionException</code> (<code>SQLException</code> 的子类) 里。
exception-sorter	提供 Boolean <code>isExceptionFatal(SQLException e)</code> 方法的 <code>org.jboss.jca.adapters.jdbc.ExceptionSorter</code> 实例。这个方法检验异常是否作为 <code>connectionErrorOccurred</code> 消息传播到所有的 <code>javax.resource.spi.ConnectionEventListener</code> 实例上。

表 6.10. 超时参数

参数	描述
use-try-lock	使用 <code>tryLock()</code> 而不是 <code>lock()</code> 。在指定秒数内试图获取锁，而不是在锁不可用时立即失败。默认值为 60 秒。如果超时为 5 分钟，则应设置 <code><use-try-lock>300</use-try-lock></code> 。
blocking-timeout-millis	等待连接时阻塞的最长时间（毫秒）。超过这个时间后，异常将被抛出。这只是在等待连接许可时阻塞，如果创建新连接花费很长时间并不会抛出异常。默认值为 30000 ，也就是 30 秒。
idle-timeout-minutes	在空闲连接关闭前的最长等待时间（分钟）。实际的最长时间取决于 <code>idleRemover</code> 扫描时间，它是任何池的最小 idle-timeout-minutes 的一半。
set-tx-query-timeout	是否根据事务超时前剩下的时间设置查询超时。如果没有事务存在则使用任何配置好的查询超时时间。默认为 false 。
query-timeout	查询的超时时间（秒）。默认是无超时。
allocation-retry	在抛出异常前，重新尝试分配连接的次数。默认为 0 ，异常将在第一次失败时抛出。
allocation-retry-wait-millis	在重新分配连接前应等待的时间（毫秒）。默认值是 5000 ，也就是 5 秒。

参数	描述
xa-resource-timeout	如果为非零值，这个值将传递给 XAResource.setTransactionTimeout 方法。

表 6.11. 语句参数

参数	描述
track-statements	<p>当连接返回池且语句返回到 prepared 语句缓存时是否检测未关闭的语句。如果为 false，则不会对语句进行追踪。</p> <p>有效值</p> <ul style="list-style-type: none"> • true:: 对语句和结果集进行跟踪，如果没有关闭则提示警告。 • false: 对语句和结果集都不进行跟踪。 • nowarn: 语句将被跟踪但不会发出警告。这是默认设置。
prepared-statement-cache-size	每个连接的 prepared 语句的个数，存在于 Least Recently Used (LRU) 缓存里。
share-prepared-statements	是否两次请求相同的底层 prepared 语句而不关闭它。默认是 false 。

表 6.12. 恢复参数

参数	描述
recover-credential	安全域用于恢复的用户名/密码对。
recover-plugin	用于恢复的 org.jboss.jca.core.spi.recoveryRecoveryPlugin 类的实现。

[提交 bug 报告](#)

6.6.2. 数据源连接 URL

表 6.13. 数据源连接 URL

数据源	数据源 URL
-----	---------

数据源	数据源 URL
PostgreSQL	<code>jdbc:postgresql://SERVER_NAME:PORT/DATABASE_NAME</code>
MySQL	<code>jdbc:mysql://SERVER_NAME:PORT/DATABASE_NAME</code>
Oracle	<code>jdbc:oracle:thin:@ORACLE_HOST:PORT:ORACLE_SID</code>
IBM DB2	<code>jdbc:db2://SERVER_NAME:PORT/DATABASE_NAME</code>
Microsoft SQLServer	<code>jdbc:microsoft:sqlserver://SERVER_NAME:PORT;DatabaseName=DATABASE_NAME</code>

[提交 bug 报告](#)

6.6.3. 数据源扩展

数据源部署可以使用 JDBC 资源适配器里的几个扩展来改进连接检验，并检查异常是否应该重新建立连接。这些扩展是：

表 6.14. 数据源扩展

数据源扩展	配置参数	描述
<code>org.jboss.jca.adapters.jdbc.spi.ExceptionSorter</code>	<code><exception-sorter></code>	检查 <code>SQLException</code> 对于抛出它的连接是否是毁灭性的
<code>org.jboss.jca.adapters.jdbc.spi.StaleConnection</code>	<code><stale-connection-checker></code>	将过时的 <code>SQLExceptions</code> 包裹在 <code>org.jboss.jca.adapters.jdbc.StaleConnectionException</code>
<code>org.jboss.jca.adapters.jdbc.spi.ValidConnection</code>	<code><valid-connection-checker></code>	检查连接是否有效，能为应用程序所用。

JBoss EAP 6 也为几个受支持的数据库实现了这些扩展。

扩展实现

通用

- `org.jboss.jca.adapters.jdbc.extensions.novendor.NullExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.novendor.NullStaleConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.novendor.NullValidConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker`

PostgreSQL

- `org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker`

MySQL

- `org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLReplicationValidConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker`

IBM DB2

- `org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker`

Sybase

- `org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker`

Microsoft SQLServer

- `org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker`

Oracle

- `org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter`
- `org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker`
- `org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker`

[提交 bug 报告](#)

6.6.4. 查看数据源统计

您可以用下列命令从定义的数据源里查看 **jdbc** 和 **pool** 的统计信息：

过程 6.11.

- `/subsystem=datasources/data-source=ExampleDS/statistics=jdbc:read-resource(include-runtime=true)`
- `/subsystem=datasources/data-source=ExampleDS/statistics=pool:read-resource(include-runtime=true)`



注意

确保您指定了 `include-runtime=true` 参数，因为所有统计都是 runtime 信息。默认值为 `false`。

[提交 bug 报告](#)

6.6.5. 数据源统计

核心统计信息

下表包含受支持的数据源核心统计信息列表：

表 6.15. 核心统计信息

名称	描述
ActiveCount	活动连接的数量。每个连接都是正在被应用程序使用或是在池里备用。
AvailableCount	池里可用连接的数量。
AverageBlockingTime	获取池里排他锁时阻塞的平均时间。单位为毫秒。
AverageCreationTime	创建连接所花费的平均时间。单位为毫秒。
CreatedCount	创建的连接数量。
DestroyedCount	销毁的连接数量。
InUseCount	正在使用的连接的数量。
MaxCreationTime	创建连接所花费的最长时间。单位为毫秒。
MaxUsedCount	使用的连接的最大数目。
MaxWaitCount	同一时间等待连接的请求的最大数目。
MaxWaitTime	等待池里排他锁所花费的最长时间。
TimedOut	超时连接的数量。
TotalBlockingTime	等待池里排他锁总共所花费的时间。单位为毫秒。
TotalCreationTime	创建连接总共所花费的时间。单位为毫秒。

名称	描述
WaitCount	需要等待连接的请求的数量。

JDBC 统计信息

下表包含受支持的数据源 JDBC 统计信息列表：

表 6.16. JDBC 统计信息

名称	描述
PreparedStatementCacheAccessCount	语句缓存被访问的次数。
PreparedStatementCacheAddCount	添加到语句缓存里的语句数量。
PreparedStatementCacheCurrentSize	目前缓存在语句缓存里的 prepared 和可调用的语句的数量。
PreparedStatementCacheDeleteCount	从缓存里丢弃的语句的数量。
PreparedStatementCacheHitCount	语句在缓存里被使用的次数。
PreparedStatementCacheMissCount	对缓存里语句的请求无法被满足的次数。

您可以用下列命令的何合适版本启用 **Core** 和 **JDBC** 的统计信息：

- ```
/subsystem=datasources/data-source=ExampleDS/statistics=pool:write-attribute(name=statistics-enabled,value=true)
```
- ```
/subsystem=datasources/data-source=ExampleDS/statistics=jdbc:write-attribute(name=statistics-enabled,value=true)
```

[提交 bug 报告](#)

6.7. 数据源示例

6.7.1. PostgreSQL 数据源示例

例 6.10.

下面的例子是一个 PostgreSQL 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```
<datasources>
  <datasource jndi-name="java:jboss/PostgresDS" pool-name="PostgresDS">
    <connection-
url>jdbc:postgresql://localhost:5432/postgresdb</connection-url>
    <driver>postgresql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidCon
nectionChecker"></valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptio
nSorter"></exception-sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="postgresql" module="org.postgresql">
      <xa-datasource-class>org.postgresql.xa.PGXADatasource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

下面是用于上面的 PostgreSQL 数据源的 `module.xml` 文件示例。

```
<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
  <resources>
    <resource-root path="postgresql-9.1-902.jdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[提交 bug 报告](#)

6.7.2. PostgreSQL XA 数据源示例

例 6.11.

下面的例子是一个 PostgreSQL XA 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```
<datasources>
  <xa-datasource jndi-name="java:jboss/PostgresXADS" pool-
```

```

name="PostgresXADS">
  <driver>postgresql</driver>
  <xa-datasource-property name="ServerName">localhost</xa-datasource-
property>
  <xa-datasource-property name="PortNumber">5432</xa-datasource-
property>
  <xa-datasource-property name="DatabaseName">postgresdb</xa-
datasource-property>
  <security>
    <user-name>admin</user-name>
    <password>admin</password>
  </security>
  <validation>
    <background-validation>true</background-validation>
    <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidCon
nectionChecker">
      </valid-connection-checker>
    <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptio
nSorter">
      </exception-sorter>
    </validation>
  </xa-datasource>
</drivers>
  <driver name="postgresql" module="org.postgresql">
    <xa-datasource-class>org.postgresql.xa.PGXADatasource</xa-
datasource-class>
  </driver>
</drivers>
</datasources>

```

下面是用于上面的 PostgreSQL XA 数据源的 `module.xml` 文件示例。

```

<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
  <resources>
    <resource-root path="postgresql-9.1-902.jdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[提交 bug 报告](#)

6.7.3. MySQL 数据源示例

例 6.12.

下面的例子是一个 MySQL 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```

<datasources>
  <datasource jndi-name="java:jboss/MySQLDS" pool-name="MySQLDS">

```

```

<connection-url>jdbc:mysql://mysql-
localhost:3306/jbossdb</connection-url>
<driver>mysql</driver>
<security>
  <user-name>admin</user-name>
  <password>admin</password>
</security>
<validation>
  <background-validation>true</background-validation>
  <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionC
hecker"></valid-connection-checker>
  <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"
></exception-sorter>
</validation>
</datasource>
<drivers>
  <driver name="mysql" module="com.mysql">
    <xa-datasource-
class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-datasource-
class>
  </driver>
</drivers>
</datasources>

```

下面是用于上面的 MySQL 数据源的 `module.xml` 文件示例。

```

<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.0.8-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[提交 bug 报告](#)

6.7.4. MySQL XA 数据源示例

例 6.13.

下面的例子是一个 MySQL XA 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```

<datasources>
  <xa-datasource jndi-name="java:jboss/MysqlXADS" pool-
name="MysqlXADS">
    <driver>mysql</driver>
    <xa-datasource-property name="ServerName">localhost</xa-datasource-
property>

```

```

    <xa-datasource-property name="DatabaseName">mysqlldb</xa-datasource-
property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionC
hecker"></valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter"
></exception-sorter>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="mysql" module="com.mysql">
      <xa-datasource-
class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-datasource-
class>
    </driver>
  </drivers>
</datasources>

```

下面是用于上面的 MySQL XA 数据源的 `module.xml` 文件示例。

```

<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.0.8-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[提交 bug 报告](#)

6.7.5. Oracle 数据源示例



注意

在 Oracle 数据源 10.2 之前的版本里，`<no-tx-separate-pools/>` 参数是必需的，因为非事务性和事务性连接的混合可能导致错误。对于某些应用程序来说，这个参数已不再是必需的了。

例 6.14.

下面的例子是一个 Oracle 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```

<datasources>

```

```

<datasource jndi-name="java:/OracleDS" pool-name="OracleDS">
  <connection-url>jdbc:oracle:thin:@localhost:1521:XE</connection-
url>
  <driver>oracle</driver>
  <security>
    <user-name>admin</user-name>
    <password>admin</password>
  </security>
  <validation>
    <background-validation>true</background-validation>
    <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectio
nChecker"></valid-connection-checker>
    <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectio
nChecker"></stale-connection-checker>
    <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorte
r"></exception-sorter>
  </validation>
</datasource>
<drivers>
  <driver name="oracle" module="com.oracle">
    <xa-datasource-
class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>

```

下面是用于上面的 Oracle 数据源的 `module.xml` 文件示例。

```

<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc6.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[提交 bug 报告](#)

6.7.6. Oracle XA 数据源示例



注意

在 Oracle 数据源 10.2 之前的版本里，`<no-tx-separate-pools/>` 参数是必需的，因为非事务性和事务性连接的混合可能导致错误。对于某些应用程序来说，这个参数已不再是必需的了。

重要

下面的设置必须应用于访问 Oracle XA 数据源的用户以使 XA 恢复可以正常操作。**user** 的值是连接 JBoss 到 Oracle 的用户：

- GRANT SELECT ON sys.dba_pending_transactions TO user;
- GRANT SELECT ON sys.pending_trans\$ TO user;
- GRANT SELECT ON sys.dba_2pc_pending TO user;
- GRANT EXECUTE ON sys.dbms_xa TO user; (If using Oracle 10g R2 (patched) or Oracle 11g)

OR

GRANT EXECUTE ON sys.dbms_system TO user; (If using an unpatched Oracle version prior to 11g)

例 6.15.

下面的例子是一个 Oracle XA 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```
<datasources>
  <xa-datasource jndi-name="java:/XAOracleDS" pool-name="XAOracleDS">
    <driver>oracle</driver>
    <xa-datasource-property name="URL">jdbc:oracle:oci8:@tc</xa-
datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
      <no-tx-separate-pools />
    </xa-pool>
    <validation>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectio
nChecker"></valid-connection-checker>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectio
nChecker"></stale-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorte
r"></exception-sorter>
    </validation>
  </xa-datasource>
</drivers>
  <driver name="oracle" module="com.oracle">
    <xa-datasource-
class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
```

```

    </driver>
  </drivers>
</datasources>

```

下面是用于上面的 Oracle XA 数据源的 `module.xml` 文件示例。

```

<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc6.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[提交 bug 报告](#)

6.7.7. Microsoft SQLServer 数据源示例

例 6.16.

下面的例子是一个 Microsoft SQLServer 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```

<datasources>
  <datasource jndi-name="java:/MSSQLDS" pool-name="MSSQLDS">
    <connection-
url>jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=MyDatabase</c
onnection-url>
    <driver>sqlserver</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSQLValidConnectionC
hecker"></valid-connection-checker>
    </validation>
  </datasource>
  <drivers>
    <driver name="sqlserver" module="com.microsoft">
      <xa-datasource-
class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasource-
class>
    </driver>
  </drivers>
</datasources>

```

下面是用于上面的 Microsoft SQLServer 数据源的 `module.xml` 文件示例。

```

<module xmlns="urn:jboss:module:1.1" name="com.microsoft">

```

```

<resources>
  <resource-root path="sqljdbc4.jar"/>
</resources>
<dependencies>
  <module name="javax.api"/>
  <module name="javax.transaction.api"/>
</dependencies>
</module>

```

[提交 bug 报告](#)

6.7.8. Microsoft SQLServer XA 数据源示例

例 6.17.

下面的例子是一个 Microsoft SQLServer XA 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```

<datasources>
  <xa-datasource jndi-name="java:/MSSQLXADS" pool-name="MSSQLXADS">
    <driver>sqlserver</driver>
    <xa-datasource-property name="ServerName">localhost</xa-datasource-
property>
    <xa-datasource-property name="DatabaseName">mssqldb</xa-datasource-
property>
    <xa-datasource-property name="SelectMethod">cursor</xa-datasource-
property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <validation>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionC
hecker"></valid-connection-checker>
    </validation>
  </xa-datasource>
</drivers>
  <driver name="sqlserver" module="com.microsoft">
    <xa-datasource-
class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasource-
class>
  </driver>
</drivers>
</datasources>

```

下面是用于上面的 Microsoft SQLServer XA 数据源的 `module.xml` 文件示例。

```

<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>

```



```

    <resource-root path="sqljdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[提交 bug 报告](#)

6.7.9. IBM DB2 数据源示例

例 6.18.

下面的例子是一个 IBM DB2 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```

<datasources>
  <datasource jndi-name="java:/DB2DS" pool-name="DB2DS">
    <connection-url>jdbc:db2:ibmdb2db</connection-url>
    <driver>ibmdb2</driver>
    <pool>
      <min-pool-size>0</min-pool-size>
      <max-pool-size>50</max-pool-size>
    </pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionCheck
er"></valid-connection-checker>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionCheck
er"></stale-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"></e
xception-sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="ibmdb2" module="com.ibm">
      <xa-datasource-class>com.ibm.db2.jdbc.DB2XADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>

```

下面是用于上面的 IBM DB2 数据源的 `module.xml` 文件示例。

```

<module xmlns="urn:jboss:module:1.1" name="com.ibm">

```

```

<resources>
  <resource-root path="db2jcc4.jar"/>
</resources>
<dependencies>
  <module name="javax.api"/>
  <module name="javax.transaction.api"/>
</dependencies>
</module>

```

[提交 bug 报告](#)

6.7.10. IBM DB2 XA 数据源示例

例 6.19.

下面的例子是一个 IBM DB2 XA 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```

<datasources>
  <xa-datasource jndi-name="java:/DB2XADS" pool-name="DB2XADS">
    <driver>ibmdb2</driver>
    <xa-datasource-property name="DatabaseName">ibmdb2db</xa-
datasource-property>
    <xa-datasource-property name="ServerName">hostname</xa-datasource-
property>
    <xa-datasource-property name="PortNumber">port</xa-datasource-
property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <validation>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionCheck
er"></valid-connection-checker>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionCheck
er"></stale-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"></e
xception-sorter>
    </validation>
    <recovery>
      <recover-plugin class-
name="org.jboss.jca.core.recovery.ConfigurableRecoveryPlugin">
        <config-property name="EnableIsValid">false</config-property>
        <config-property name="IsValidOverride">false</config-property>
        <config-property name="EnableClose">false</config-property>
      </recover-plugin>
    </recovery>
  </xa-datasource>
</datasources>

```

```

</xa-datasource>
<drivers>
  <driver name="ibmdb2" module="com.ibm">
    <xa-datasource-class>com.ibm.db2.jcc.DB2XADataSource</xa-
datasource-class>
  </driver>
</drivers>
</datasources>

```

下面是用于上面的 IBM DB2 XA 数据源的 `module.xml` 文件示例。

```

<module xmlns="urn:jboss:module:1.1" name="com.ibm">
  <resources>
    <resource-root path="db2jcc4.jar"/>
    <resource-root path="db2jcc_license_cisuz.jar"/>
    <resource-root path="db2jcc_license_cu.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>

```

[提交 bug 报告](#)

6.7.11. Sybase 数据源示例

例 6.20.

下面的例子是一个 Sybase 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```

<datasources>
  <datasource jndi-name="java:jboss/SybaseDB" pool-name="SybaseDB"
enabled="true">
    <connection-url>jdbc:sybase:Tds:localhost:5000/DATABASE?
JCONNECT_VERSION=6</connection-url>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectio
nChecker"></valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorte
r"></exception-sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="sybase" module="com.sybase">
      <datasource-
class>com.sybase.jdbc4.jdbc.SybDataSource</datasource-class>

```

```

        <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-
datasource-class>
    </driver>
</drivers>
</datasources>

```

下面是用于上面的 Sybase 数据源的 `module.xml` 文件示例。

```

<module xmlns="urn:jboss:module:1.1" name="com.sybase">
    <resources>
        <resource-root path="jconn2.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>

```

[提交 bug 报告](#)

6.7.12. Sybase XA 数据源示例

例 6.21.

下面的例子是一个 Sybase XA 数据源配置。这个数据源已被启用，用户已经添加并已设置了检验选项。

```

<datasources>
    <xa-datasource jndi-name="java:jboss/SybaseXADS" pool-
name="SybaseXADS" enabled="true">
        <xa-datasource-property name="NetworkProtocol">Tds</xa-datasource-
property>
        <xa-datasource-property name="ServerName">myserver</xa-datasource-
property>
        <xa-datasource-property name="PortNumber">4100</xa-datasource-
property>
        <xa-datasource-property name="DatabaseName">mydatabase</xa-
datasource-property>
        <security>
            <user-name>admin</user-name>
            <password>admin</password>
        </security>
        <validation>
            <background-validation>true</background-validation>
            <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectio
nChecker"></valid-connection-checker>
            <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorte
r"></exception-sorter>
        </validation>
    </xa-datasource>
</drivers>

```

```
<driver name="sybase" module="com.sybase">
  <datasource-
class>com.sybase.jdbc4.jdbc.SybDataSource</datasource-class>
  <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-
datasource-class>
  </driver>
</drivers>
</datasources>
```

下面是用于上面的 Sybase XA 数据源的 `module.xml` 文件示例。

```
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
  <resources>
    <resource-root path="jconn2.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

[提交 bug 报告](#)

第 7 章 配置模块

7.1. 介绍

7.1.1. 模块

模块是用于类加载和依赖关系管理的类的逻辑组。JBoss EAP 6 使用两种模块类型，有时称为静态和动态模块。然而，这两者的区别只是它们打包的方式。所有模块都提供相同的功能。

静态模块

静态模块是在应用服务器的 **EAP_HOME/modules/** 目录里进行预定义的。每个子目录都代表一个模块并包含一个配置文件 (**module.xml**) 或所需的 JAR 文件。模块的名称是在 **module.xml** 文件里定义的。所有应用服务器提供的 API 都是作为静态模块提供的，包括 Java EE API 以及其他 API (如 Jboss Logging)。

例 7.1. module.xml 文件示例

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.1.15.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

模块名 **com.mysql** 应该匹配这个模块的目录结构。

JBoss EAP 里提供的模块位于 **JBOSS_HOME/modules** 里的 **system** 目录。这使它们和第三方体的模块隔离开来。

Red Hat 提供的任何高于 JBoss EAP 6.1 或更高版本的分层产品也会安装它们的模块到 **system** 目录里。

如果许多使用相同第三方库的应用程序部署在相同的服务器上，那么创建自定义静态模块就很有用。不是将这些库捆绑到每个应用程序，而是由 JBoss 管理员创建和安装包含这些库的模块。然后应用程序就可以声明对自定义静态模块的显性依赖关系。

用户必须确保使用一个目录对应一个模块的格式将自定义的模块安装至 **JBOSS_HOME/modules** 目录。这可以确保系统加载已存在于 **system** 目录的模块的自定义版本，而不是系统附带的版本。因此，用户提供的模块将优先于系统模块。

如果您使用 **JBOSS_MODULE_PATH** 环境变量来修改 JBoss EAP 搜索模块的位置，那么产品将查找指定的位置中的 **system** 子目录结构。**system** 结构必须存在于用 **JBOSS_MODULE_PATH** 指定的位置中的某处。

动态模块

应用服务器为每个 JAR 或 WAR 部署（或 EAR 里的子部署）创建和加载动态模块。动态模块的名称源自部署的归档的名称。因为部署是作为模块加载的，它们可以配置依赖关系并被其他部署作为依赖关系使用。

模块只有在需要时才会加载。这通常只在具有显性或隐性依赖关系的应用程序部署时发生。

[提交 bug 报告](#)

7.1.2. 全局模块

全局模块（Global Module）是 JBoss EAP 6 作为每个应用程序的依赖关系而提供的模块。将任何模块添加到应用服务器的全局模块列表里都可以使其成为全局模块，你并不需要修改模块。

[提交 bug 报告](#)

7.1.3. 模块的依赖关系

模块依赖关系是某个模块要求另外一个模块的类行使功能的声明。模块可以声明对多个模块的依赖关系。当应用服务器加载一个模块时，模块类加载器将解析该模块的依赖关系并从每个依赖关系中添加类到 class path 里。如果无法找到指定的依赖关系，这个模块将加载失败。

部署的应用程序（JAR 和 WAR）将被加载为动态模块并使用依赖关系来访问 JBoss EAP 6 提供的 API。

依赖关系的类型有两种：explicit（显性的）和 implicit（隐性的）

显性依赖关系是由开发人员声明的。静态的模块可以在 modules.xml 文件里声明依赖关系。动态模块可以在 MANIFEST.MF 或 jboss-deployment-structure.xml 里声明依赖关系。

您可以指定显性的依赖关系为可选的。无法加载可选依赖关系不会导致模块无法加载。然而，如果依赖关系以后变成可用时，它将无法添加到模块的 Class path 里。模块加载时依赖关系就必须是可用的。

隐性的依赖关系在某些条件满足或在部署里发现元数据时自动由应用服务器添加。JBoss EAP 6 附带的 Java EE 6 API 是通过检测部署里的隐性依赖关系而添加的模块的示例。

我们可以配置部署排除特定的隐性依赖关系。这是通过 jboss-deployment-structure.xml 部署描述符文件来完成的。这常常发生在当应用程序捆绑某个版本的库文件，而应用服务器将试图将其添加为隐性依赖关系的时候。

模块的 Class path 包含自己的类及其直接的依赖关系。模块无法访问其中一个依赖关系的依赖关系类。然而，模块可以指定导出某个显性的依赖关系。导出的依赖关系将提供给任何依赖于导出它的模块的模块。

例 7.2. 模块依赖关系

模块 A 依赖于模块 B 而模块 B 依赖于模块 C。模块 A 可以访问模块 B 的类，模块 B 可以访问模块 C 的类。模块 A 无法访问模块 C 的类。

- 模块 A 声明了对模块 C 的显性依赖关系，或者
- 模块 B 导出它对模块 C 的依赖关系。

[提交 bug 报告](#)

7.1.4. 子部署类加载器的隔离

EAR 里的每个子部署都是一个动态模块，它有自己的类加载器。在默认情况下，子部署可以访问其他子部署的资源。

如果子部署不应该访问其他子部署的资源（要求严格的子部署隔离），那么就可以启用隔离。

[提交 bug 报告](#)

7.2. 对所有的部署禁用子部署模块隔离（SUB-DEPLOYMENT MODULE ISOLATION）

这个任务展示了服务器管理员在应用服务期上如何禁用子部署模块隔离。这会影响到所有的部署。



警告

这个任务要求您编辑服务器的 XML 配置文件。编辑前这个服务器必须先暂停，这是临时措施，最终版本的管理工具将会支持这种配置。

1. 停止服务器

暂停 JBoss EAP 6 服务器。

2. 打开服务器配置文件

在文本编辑器里打开服务器配置文件。

对于受管域和独立服务器这个文件是不同的。此外也可能使用非默认的位置和文件名称。对于受管域和独立服务器，默认的配置文件的名称分别是 **domain/configuration/domain.xml** 和 **standalone/configuration/standalone.xml**。

3. 定位 EE 子系统配置

在配置文件里找到 EE 子系统配置元素。配置文件的 **<profile>** 元素包含了几个子系统元素。EE 子系统元素的命名空间是 **urn:jboss:domain:ee:1.1**。

```
<profile>
    ...
    <subsystem xmlns="urn:jboss:domain:ee:1.1" />
    ...
```

默认的配置有一个自闭合的标签，但自定义的配置可能有单独的开和合标签（里面可能还有其他元素），如：

```
<subsystem xmlns="urn:jboss:domain:ee:1.1" ></subsystem>
```

4. 如果需要则替换自闭合的标签

如果 EE Subsystem 元素是一个单个的自闭合标签，那么请用合适的开和合标签来替换，如：

```
<subsystem xmlns="urn:jboss:domain:ee:1.1" ></subsystem>
```


5. 添加 ear-subdeployments-isolated 元素

将 **ear-subdeployments-isolated** 元素添加为 EE Subsystem 元素的子元素并添加 **false** 内容，如：

```
<subsystem xmlns="urn:jboss:domain:ee:1.1" ><ear-subdeployments-isolated>false</ear-subdeployments-isolated></subsystem>
```

6. 期待服务器

重新启动 JBoss EAP 6 服务器以使用新的配置运行。

结果：

服务器现在对于所有部署都禁用了 Subdeployment Module Isolation。

[提交 bug 报告](#)

7.3. 添加模块到所有部署里

这个任务展示了 JBoss 管理员如何定义全局模块列表。

必须具备的条件

1. 您必须知道要配置为全局模块的模块的名称。关于 JBoss EAP 6 附带的静态模块的列表，请参考 [第 7.5.1 节“包括的模块”](#)。如果这个模块是另外一个部署，请参考 [第 7.5.2 节“动态模块命名”](#) 来确定模块名。

过程 7.1. 添加模块到全局模块列表里

1. 登陆到管理控制台。 [第 3.4.2 节“登录到管理控制台”](#)
2. 进入 **EE Subsystem** 面板。
 - a. 从控制台顶部选择 **Configuration** 标签页。
 - b. 仅用于域模式
 - i. 从左上角的下拉菜单里选择合适的配置集。
 - c. 展开控制台左侧的 **Subsystems** 菜单。
 - d. 从控制台左侧的菜单里选择 **Container** → **EE**。
3. 点击 **Subsystem Defaults** 部分的 **Add** 按钮。**Create Module** 对话框将会出现。
4. 输入模块名以及模块 slot（可选）。
5. 点击 **Save** 按钮来添加新的全局模块，或者点击 **Cancel** 链接中止。
 - 如果您点击了 **Save** 按钮，对话框将关闭而指定的模块将被添加到全局模块列表里。
 - 如果您点击了 **Cancel**，对话框将关闭且不会保存任何修改。

结果

添加到全局模块列表的模块将作为依赖关系添加到每个部署里。

[提交 bug 报告](#)

7.4. 定义外部 JBOSS MODULES 目录

介绍

在默认情况下，JBoss EAP 查找 *EAP_HOME/modules/* 目录里的模块。通过定义 **JBOSS_MODULEPATH** 环境变量或在启动配置文件里设置变量，您可以指引 JBoss EAP 查找一个或多个外部目录。本节描述了这两个方法。

过程 7.2. 设置 JBOSS_MODULEPATH 环境变量

- 要指定一个或更多的外部模块目录，请定义环境变量 **JBOSS_MODULEPATH**。

在 Linux 里，请使用冒号来隔开目录列表。例如：

```
export
JBOSS_MODULEPATH=EAP_HOME/modules/:/home/username/external/modules/d
irectory/
```

在 Windows 里，请使用分号来隔开目录列表。例如：

```
SET JBOSS_MODULEPATH=EAP_HOME\modules\;D:\JBoss-Modules\
```

过程 7.3. 在 Startup 配置文件里设置 JBOSS_MODULEPATH 环境变量

- 如果您不愿意设置全局环境变量，您可以在 JBoss EAP 启动配置文件里设置 **JBOSS_MODULEPATH** 变量。如果您运行的是独立服务器，配置文件是 *EAP_HOME/bin/standalone.conf*；如果服务器运行在受管域里，配置文件是 *EAP_HOME/bin/domain.conf*。

下面是在 *standalone.conf* 文件里设置 **JBOSS_MODULEPATH** 变量的命令示例。

```
JBOSS_MODULEPATH="EAP_HOME/modules/:/home/username/external/modules/
directory/"
```

[提交 bug 报告](#)

7.5. 参考

7.5.1. 包括的模块

您可以在客户门户找到 JBoss EAP 6 包含的模块列表以及是否受支持：<https://access.redhat.com/articles/1122333>。

[提交 bug 报告](#)

7.5.2. 动态模块命名

所有部署都被 JBoss EAP 6 加载为模块并按照下列规格进行命名。

1. WAR 和 JAR 文件的部署是用下列格式命名的：

■

```
deployment.DEPLOYMENT_NAME
```

例如，**inventory.war** 和 **store.jar** 的模块名分别是 **deployment.inventory.war** 和 **deployment.store.jar**。

2. EAR 里的子部署是用下列格式命名的：

```
deployment.EAR_NAME.SUBDEPLOYMENT_NAME
```

例如，**accounts.ear** 里的 **reports.war** 子部署的模块名将是 **deployment.accounts.ear.reports.war**。

[提交 bug 报告](#)

第 8 章 JSVC

8.1. 介绍

8.1.1. 关于 Jsvc

Jsvc 是一系列允许 Java 应用程序作为后台服务运行在 Unix 或类 Unix 平台里的库和应用程序。它允许应用程序作为特权用户执行操作，然后将其身份切换到非特权用户。

Jsvc 使用三个进程：launcher、controller 和 controlled 进程。controlled 进程也是主 Java 线程。如果 JVM 崩溃，controller 进程将在 60 秒内重启它。Jsvc 是 JBoss EAP 的守护进程，它只能由特权用户启动。



注意

Jsvc 仅用在 Red Hat Enterprise Linux、Solaris 和 HP-UX 里。关于 Microsoft Windows 上的类似功能，请使用 Red Hat 客户门户里 **Native Utilities for Windows Server** 的 `prunsrv.exe`。

[提交 bug 报告](#)

8.1.2. 用 Jsvc 启动和停止 JBoss EAP

根据操作模式的不同，用 Jsvc 启动和停止 JBoss EAP 的说明也会不同。请注意如果 JBoss EAP 运行在域模式里，Jsvc 只会负责域控制器的处理过程。不管您使用哪个命令通过 Jsvc 启动 JBoss EAP，它都必须用特权用户来运行。

必须具备的条件

- 如果 JBoss EAP 6 是用 ZIP 方式安装的：
 - 从 Red Hat 客户门户下载并安装 *Native Utilities* 软件包。请参考《安装指南》里的『安装 Native 组件和 Native 工具 (ZIP/安装程序)』章节。
 - 创建将运行 JBoss EAP 6 实例的用户帐号。用来启动和停止服务器的帐号必须拥有读取和写入安装 JBoss EAP 的目录的权限。
- 如果 JBoss EAP 是用 RPM 模式安装的，请安装 *apache-commons-daemon-jsvc-eap6* 软件包。请参考《安装指南》里的『安装 Native 组件和 Native 工具 (RPM 安装)』章节。

下面的命令是在独立或域模式下启动和停止 JBoss EAP。请注意，根据安装 Jsvc 的方法的不同，文件的位置也会不同。请用下表来确定使用哪些文件来解析命令里的变量。

独立模式

下面的说明是关于在独立模式下启动和停止 JBoss EAP 的。

表 8.1. ZIP 安装模式下的 Jsvc 文件位置 - 独立模式

参考文件说明	文件位置
EAP-HOME	<code>\${eap-installation-location}/jboss-eap-\${version}</code>

参考文件说明	文件位置
JSVC-BIN	EAP_HOME/modules/system/layers/base/native/sbin/jsvc
JSVC-JAR	EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar
CONF-DIR	EAP_HOME/standalone/configuration
LOG-DIR	EAP_HOME/standalone/log

表 8.2. RPM 安装模式下的 Jsvc 文件位置 - 独立模式

参考文件说明	文件位置
EAP-HOME	/usr/share/jbossas
JSVC-BIN	/usr/bin/jsvc-eap6/jsvc
JSVC-JAR	EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar
CONF-DIR	/etc/jbossas/standalone
LOG-DIR	/var/log/jbossas/standalone

以独立模式启动 JBoss EAP

- ```

JSVC_BIN \
-outfile LOG_DIR/jsvc.out.log \
-errfile LOG_DIR/jsvc.err.log \
-pidfile LOG_DIR/jsvc.pid \
-user jboss \
-D[Standalone] -XX:+UseCompressedOops -Xms1303m \
-Xmx1303m -XX:MaxPermSize=256m \
-Djava.net.preferIPv4Stack=true
-Djboss.modules.system.pkgs=org.jboss.byteman \
-Djava.awt.headless=true \
-Dorg.jboss.boot.log.file=LOG_DIR/server.log \
-Dlogging.configuration=file:CONF_DIR/logging.properties \
-Djboss.modules.policy-permissions \
-cp EAP_HOME/jboss-modules.jar:JSVC_JAR \
-Djboss.home.dir=EAP_HOME \
-Djboss.server.base.dir=EAP_HOME/standalone \
@org.jboss.modules.Main -start-method main \
-mp EAP_HOME/modules \
-jaxpmodule javax.xml.jaxp-provider \
org.jboss.as.standalone

```

### 停止以独立模式运行的 JBoss EAP

- ```

JSVC_BIN \
-stop \
-outfile LOG_DIR/jsvc.out.log \
-errfile LOG_DIR/jsvc.err.log \
-pidfile LOG_DIR/jsvc.pid \
-user jboss \
-D[Standalone] -XX:+UseCompressedOops -Xms1303m \
-Xmx1303m -XX:MaxPermSize=256m \
-Djava.net.preferIPv4Stack=true \
-Djboss.modules.system.pkgs=org.jboss.byteman \
-Djava.awt.headless=true \
-Dorg.jboss.boot.log.file=LOG_DIR/server.log \
-Dlogging.configuration=file:CONF_DIR/logging.properties \
-Djboss.modules.policy-permissions \
-cp EAP_HOME/jboss-modules.jar:JSVC_JAR \
-Djboss.home.dir=EAP_HOME \
-Djboss.server.base.dir=EAP_HOME/standalone \
@org.jboss.modules.Main -start-method main \
-mp EAP_HOME/modules \
-jaxpmodule javax.xml.jaxp-provider \
org.jboss.as.standalone

```

域模式

下面的说明是关于在域模式下启动和停止 JBoss EAP 的。请注意对于域模式，您必须用 Java 主目录替换 `JAVA_HOME` 变量。

表 8.3. ZIP 安装模式下的 Jsvc 文件位置 - 域模式

参考文件说明	文件位置
EAP-HOME	<code>\${eap-installation-location}/jboss-eap-\${version}</code>
JSVC-BIN	<code>EAP_HOME/modules/system/layers/base/native/sbin/jsvc</code>
JSVC-JAR	<code>EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar</code>
CONF-DIR	<code>EAP_HOME/domain/configuration</code>
LOG-DIR	<code>EAP_HOME/domain/log</code>

表 8.4. RPM 安装模式下的 Jsvc 文件位置 - 域模式

参考文件说明	文件位置
EAP-HOME	<code>/usr/share/jbossas</code>
JSVC-BIN	<code>/usr/bin/jsvc-eap6/jsvc</code>
JSVC-JAR	<code>EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar</code>

参考文件说明	文件位置
CONF-DIR	/etc/jbossas/domain
LOG-DIR	/var/log/jbossas/domain

以域模式启动 JBoss EAP

- ```

JSVC_BIN \
-outfile LOG_DIR/jsvc.out.log \
-errfile LOG_DIR/jsvc.err.log \
-pidfile LOG_DIR/jsvc.pid \
-user jboss \
-nodetach -D"[Process Controller]" -server -Xms64m \
-Xmx512m -XX:MaxPermSize=256m \
-Djava.net.preferIPv4Stack=true \
-Djboss.modules.system.pkgs=org.jboss.byteman \
-Djava.awt.headless=true \
-Dorg.jboss.boot.log.file=LOG_DIR/process-controller.log \
-Dlogging.configuration=file:CONF_DIR/logging.properties \
-Djboss.modules.policy-permissions \
-cp "EAP_HOME/jboss-modules.jar:JSVC_JAR" \
org.apache.commons.daemon.support.DaemonWrapper \
-start org.jboss.modules.Main -start-method main \
-mp EAP_HOME/modules org.jboss.as.process-controller \
-jboss-home EAP_HOME -jvm $JAVA_HOME/bin/java \
-mp EAP_HOME/modules -- \
-Dorg.jboss.boot.log.file=LOG_DIR/host-controller.log \
-Dlogging.configuration=file:CONF_DIR/logging.properties \
-Djboss.modules.policy-permissions \
-server -Xms64m -Xmx512m -XX:MaxPermSize=256m \
-Djava.net.preferIPv4Stack=true \
-Djboss.modules.system.pkgs=org.jboss.byteman \
-Djava.awt.headless=true -- -default-jvm $JAVA_HOME/bin/java

```

## 停止以域模式运行的 JBoss EAP

- ```

JSVC_BIN \
-stop \
-outfile LOG_DIR/jsvc.out.log \
-errfile LOG_DIR/jsvc.err.log \
-pidfile LOG_DIR/jsvc.pid \
-user jboss \
-nodetach -D"[Process Controller]" -server -Xms64m \
-Xmx512m -XX:MaxPermSize=256m \
-Djava.net.preferIPv4Stack=true \
-Djboss.modules.system.pkgs=org.jboss.byteman \
-Djava.awt.headless=true \
-Dorg.jboss.boot.log.file=LOG_DIR/process-controller.log \
-Dlogging.configuration=file:CONF_DIR/logging.properties \
-Djboss.modules.policy-permissions \
-cp "EAP_HOME/jboss-modules.jar:JSVC_JAR" \
org.apache.commons.daemon.support.DaemonWrapper \

```

```
-start org.jboss.modules.Main -start-method main \  
-mp EAP_HOME/modules org.jboss.as.process-controller \  
-jboss-home EAP_HOME -jvm $JAVA_HOME/bin/java \  
-mp EAP_HOME/modules -- \  
-Dorg.jboss.boot.log.file=LOG_DIR/host-controller.log \  
-Dlogging.configuration=file:CONF_DIR/logging.properties \  
-Djboss.modules.policy-permissions \  
-server -Xms64m -Xmx512m -XX:MaxPermSize=256m \  
-Djava.net.preferIPv4Stack=true \  
-Djboss.modules.system.pkgs=org.jboss.byteman \  
-Djava.awt.headless=true -- -default-jvm $JAVA_HOME/bin/java
```



注意

如果 JBoss EAP 6 异常终止，如 JVM 崩溃，Jsvc 将自动重启它。如果 JBoss EAP 6 是正常终止的，Jsvc 也会随之停止。

[提交 bug 报告](#)

第 9 章 全局 VALVE

9.1. 关于 VALVE

Valve 是一个 Java 类，它在 Servlet 过滤器之前插入到应用程序的请求处理管道里。Valve 可以在将请求传递或执行其他处理（如验证或取消请求）之前修改请求。

Valve 可以在服务器或应用程序级别进行配置。唯一的区别是如何配置和打包。

- 全局 Valve 是在服务器级别配置的，它应用于所有部署在服务器里的应用程序。配置全局 Valve 的说明位于《JBoss EAP 管理和配置指南》里。
- 在应用程序级别配置的 Valve 和应用程序部署打包在一起，它只影响专门的应用程序。配置程序级别的 Valve 的说明位于《JBoss EAP 开发指南》里。

6.1.0 和以后的版本支持全局 Valve。

[提交 bug 报告](#)

9.2. 关于全局 VALVE

全局 Valve 是插入到应用程序的请求处理管道里的 Valve。在 JBoss EAP 6 里 Valve 可以通过打包或安装为静态模块来成为全局 Valve。全局 Valve 是在 web 子系统里配置的。

只有 6.1.0 和以后的版本支持全局 Valve。

关于如何配置全局 Valve 的说明，请参考《JBoss EAP 管理和配置指南》里的『全局 Valve』。章节。

[提交 bug 报告](#)

9.3. 关于 AUTHENTICATOR VALVE

Authenticator Valve 是一个验证请求的凭证的 Valve。它是 `org.apache.catalina.authenticator.AuthenticatorBase` 的一个子类并重写了 `authenticate(Request request, Response response, LoginConfig config)` 方法。

它可以用于实现其他的验证模式。

[提交 bug 报告](#)

9.4. 安装 GLOBAL VALVE

全局 Valve 必须打包和安装为 JBoss EAP 6 里的静态模块。这个任务展示了如何安装模块。

预备条件：

- Valve 必须已被创建且打包在 JAR 文件里。
- 必须为这个模块创建一个 `module.xml` 文件。

关于 `module.xml` 文件的例子，请参考 [第 7.1.1 节“模块”](#)。

过程 9.1. 安装全局模块 (Global Module)

1. 创建模块安装目录

在应用服务器的 **modules** 目录里必须创建一个安装模块的目录。

```
EAP_HOME/modules/system/layers/base/MODULENAME/main
```

```
$ mkdir -P EAP_HOME/modules/system/layers/base/MODULENAME/main
```

2. 复制文件

复制 **JAR** 和 **module.xml** 文件到步骤 1 创建的目录里。

```
$ cp MyValves.jar module.xml
EAP_HOME/modules/system/layers/base/MODULENAME/main
```

模块里声明的 **Valve** 类现在可在 **web** 子系统里进行配置。

[提交 bug 报告](#)

9.5. 配置全局 VALVE

全局 **Valve** 可以在 **web** 子系统里启用和配置。这是用 **JBoss CLI** 工具来完成的。

过程 9.2. 配置全局 Valve

1. 启用 Valve

使用 **add** 操作添加新的 **Valve** 条目。

```
/subsystem=web/valve=VALVENAME:add(module="MODULENAME",class-
name="CLASSNAME")
```

您需要指定下列值：

- **VALVENAME**，用来引用应用程序配置里的 **valve** 的名称。
- **MODULENAME**，包含要配置的 **Valve** 的模块。
- **CLASSNAME**，模块里专有 **valve** 的类名。

```
/subsystem=web/valve=clientlimiter:add(module="clientlimitermodule",
class-name="org.jboss.samplevalves.RestrictedUserAgentsValve")
```

2. 可选：指定参数

如果 **Valve** 有配置参数，请用 **add-param** 操作进行指定。

```
/subsystem=web/valve=testvalve:add-param(param-name="NAME", param-
value="VALUE")
```

```
/subsystem=web/valve=testvalve:add-param(
    param-name="restrictedUserAgents",
    param-value="^.*MS Web Services Client Protocol.*$"
)
```

Valve 已为所有部署的应用程序启用且配置好。

[提交 bug 报告](#)

第 10 章 应用程序部署

10.1. 关于应用程序部署

JBoss EAP 6 带有一系列的应用程序部署和配置选项以满足管理和开发环境的需要。对于管理员而言，管理控制台和管理 CLI 提供了理想的图形化和命令行界面来管理产品环境里的应用程序部署。对于开发人员而言，应用程序部署测试选项包含了高度可配置的文件系统部署扫描器、对 IDE 如 JBoss Developer Studio 的使用及通过 Maven 进行部署和卸载。

Administration

- **管理控制台**
 - [第 10.2.2 节 “用管理控制台启用已部署的应用程序”](#)
 - [第 10.2.3 节 “用管理控制台禁用已部署的应用程序”](#)
- **管理 CLI**
 - [第 10.3.4 节 “用管理 CLI 在受管域里部署应用程序”](#)
 - [第 10.3.2 节 “用管理 CLI 在独立服务器里部署应用程序”](#)
 - [第 10.3.5 节 “用管理 CLI 卸载受管域里的应用程序”](#)
 - [第 10.3.3 节 “用管理 CLI 卸载独立服务器里的应用程序”](#)
 - [第 10.3.1 节 “在管理 CLI 里管理应用程序的部署”](#)

开发

- **部署扫描器**
 - [第 10.5.7 节 “配置部署扫描器”](#)
 - [第 10.5.2 节 “用部署扫描器部署应用程序到独立服务器实例”](#)
 - [第 10.5.3 节 “用部署扫描器卸载独立服务器实例的应用程序”](#)
 - [第 10.5.4 节 “用部署扫描器在独立服务器实例里重新部署应用程序”](#)
 - [第 10.5.8 节 “用管理 CLI 配置部署扫描器”](#)
 - [第 10.5.6 节 “对部署扫描器属性的引用”](#)
 - [第 10.5.5 节 “对部署扫描器 Marker 文件的引用”](#)
- **Maven**
 - [第 10.6.2 节 “用 Maven 部署应用程序”](#)
 - [第 10.6.3 节 “用 Maven 卸载应用程序”](#)

[提交 bug 报告](#)

10.2. 用管理控制台进行部署

10.2.1. 在管理控制台里管理应用程序的部署

通过管理控制台部署应用程序可为你提供易于使用的图形界面。你可以马上看到部署到服务器或组里的应用程序，而且你可以按照需要禁用或删除内容库里的应用程序。

[提交 bug 报告](#)

10.2.2. 用管理控制台启用已部署的应用程序

预备条件

- [第 3.4.2 节 “登录到管理控制台”](#)
- [第 3.4.7 节 “在管理控制台里添加部署”](#)

过程 10.1. 用管理控制台启用已部署的应用程序

1. 从控制台的顶部选择 **Runtime** 标签页。
2.
 - 对于受管域，展开 **Domain** 菜单。
 - 对于独立服务器，展开 **Server** 菜单。
3. 选择 **Manage Deployments**。
4. 应用程序的部署方法会根据部署到独立服务器还是受管域而有所不同。
 - **启用独立服务器实例里的应用程序**
Available Deployments 表显示所有可用的应用程序部署及其状态。
 - a. 要启用独立服务器实例里的应用程序，请选择应用程序并点击 **En/Disable**。
 - b. 点击 **confirm** 按钮来确认应用程序将在服务器实例上启用。
 - **启用受管域里的应用程序**
Content Repository 标签页包含一个 **Available Deployment Content** 表显示所有可用的应用程序部署及其状态。
 - a. 要启用受管域里的应用程序，请选择要部署的应用程序。点击 **Available Deployment Content** 表上的 **Assign**。
 - b. 选择您要添加应用程序的服务器组并点击 **Save** 按钮完成。
 - c. 选择 **Server Groups** 标签页来查看 **Server Groups** 表。您的应用程序现在已部署到所选的服务器组了。

结果

应用程序部署在相关的服务器或服务器组。

[提交 bug 报告](#)

10.2.3. 用管理控制台禁用已部署的应用程序

预备条件

- [第 3.4.2 节 “登录到管理控制台”](#)
- [第 3.4.7 节 “在管理控制台里添加部署”](#)
- [第 10.2.2 节 “用管理控制台启用已部署的应用程序”](#)

过程 10.2. 用管理控制台禁用已部署的应用程序

1. a. 从控制台的顶部选择 **Runtime** 标签页。
 - b. ■ 对于受管域，展开 **Domain** 菜单。
 - 对于独立服务器，展开 **Server** 菜单。
 - c. 选择 **Manage Deployments**。
2. 禁用应用程序的方法会根据部署到独立服务器还是受管域而有所不同。
 - **禁用独立服务器实例里已部署的应用程序**
Available Deployments 表显示所有可用的应用程序部署及其状态。
 - a. 选择要禁用的应用程序。点击 **En/Disable** 禁用所选的应用程序。
 - b. 点击 **confirm** 按钮来确认应用程序将在服务器实例上禁用。
 - **禁用受管域里已部署的应用程序**
Manage Deployments Content 屏幕包含一个 **Content Repository** 标签页，**Available Deployment Content** 表显示所有可用的应用程序部署及其状态。
 - a. 选择 **Server Group** 标签页来查看服务器组以及部署的应用程序的状态。
 - b. 选择 **Server Group** 表里的服务器来卸载应用程序。点击 **View** 查看应用程序。
 - c. 选择应用程序并点击 **En/Disable** 禁用所选服务器的应用程序。
 - d. 点击 **confirm** 按钮来确认应用程序将在服务器实例上禁用。
 - e. 按需要对其他服务器组重复这些步骤。**Group Deployments** 表里可以确认每个服务器组里的应用程序的状态。

结果

应用程序已从相关的服务器或服务器组里卸载。

[提交 bug 报告](#)

10.3. 用管理 CLI 进行部署

10.3.1. 在管理 CLI 里管理应用程序的部署

通过管理 CLI 部署应用程序可让您用简单的命令行来创建和运行部署脚本。您可以使用脚本来配置专有的应用程序部署和管理场景。您既可以管理独立服务器实例里单个服务器的部署状态，也可以管理受管域里整个服务器网络。

[提交 bug 报告](#)

10.3.2. 用管理 CLI 在独立服务器里部署应用程序

前提条件

- [第 3.5.2 节“启动管理 CLI”](#)
- [第 3.5.4 节“用管理 CLI 连接受管服务器实例”](#)

过程 10.3. 在独立服务器里部署应用程序

- **运行 `deploy` 命令**
在管理 CLI 里，输入 `deploy` 命令及应用程序部署的位置。

```
[standalone@localhost:9999 /] deploy /path/to/test-application.war
```

请注意，成功的部署不会在 CLI 里产生任何输出。

结果

指定的应用程序已部署在独立服务器里了。

[提交 bug 报告](#)

10.3.3. 用管理 CLI 卸载独立服务器里的应用程序

前提条件

- [第 3.5.2 节“启动管理 CLI”](#)
- [第 3.5.4 节“用管理 CLI 连接受管服务器实例”](#)
- [第 10.3.2 节“用管理 CLI 在独立服务器里部署应用程序”](#)

过程 10.4. 卸载独立服务器里的应用程序

- **运行 `undeploy` 命令**
在管理 CLI 里，输入 `undeploy` 命令及应用程序部署的文件名。

```
[standalone@localhost:9999 /] undeploy test-application.war
```

请注意，成功的卸载不会在 CLI 里产生任何输出。

结果

指定的应用程序已经卸载了。

[提交 bug 报告](#)

10.3.4. 用管理 CLI 在受管域里部署应用程序

前提条件

- [第 3.5.2 节 “启动管理 CLI”](#)
- [第 3.5.4 节 “用管理 CLI 连接受管服务器实例”](#)

过程 10.5. 在受管域里部署应用程序

- **运行 deploy 命令**

在管理 CLI 里，输入 **deploy** 命令及应用程序部署的位置。如要部署到所有的服务器组，可以使用 **--all-server-groups** 参数。

```
[domain@localhost:9999 /] deploy /path/to/test-application.war --all-server-groups
```

- 或者，用 **--server-groups** 参数定义部署的专有服务器组。

```
[domain@localhost:9999 /] deploy /path/to/test-application.war --server-groups=server_group_1,server_group_2
```

请注意，成功的部署不会在 CLI 里产生任何输出。

结果

指定的应用程序现在已部署在受管域的服务器组里了。

[提交 bug 报告](#)

10.3.5. 用管理 CLI 卸载受管域里的应用程序

前提条件

- [第 3.5.2 节 “启动管理 CLI”](#)
- [第 3.5.4 节 “用管理 CLI 连接受管服务器实例”](#)
- [第 10.3.4 节 “用管理 CLI 在受管域里部署应用程序”](#)

过程 10.6. 卸载受管域里的应用程序

- **运行 undeploy 命令**

在管理 CLI 里，输入 **undeploy** 命令及应用程序部署的文件名。如果要从应用程序原来部署过的所有服务器组里卸载这个应用程序，可以使用 **--all-relevant-server-groups** 参数。

```
[domain@localhost:9999 /] undeploy test-application.war --all-relevant-server-groups
```

请注意，成功的卸载不会在 CLI 里产生任何输出。

结果

指定的应用程序已经卸载了。

[提交 bug 报告](#)

10.4. 用 HTTP API 进行部署

10.4.1. 用 HTTP API 部署应用程序

概述

应用程序可以按照下列说明通过 HTTP API 进行部署。

过程 10.7. 用 DeployDmrToJson.java 部署应用程序

1. 用 DeployDmrToJson.java 为 JSON 生成请求来部署应用程序。

例 10.1. DeployDmrToJson.java class

```
import org.jboss.dmr.ModelNode;
import java.net.URL;

public class DeployDmrToJson
{
    public static void main(String[] args) throws Exception
    {
        if(args.length < 1)
            throw new IllegalArgumentException("The first argument must
be a URL");

        URL url = new URL(args[0]);
        String[] pathElements = url.getFile().split("/");
        String name = pathElements[pathElements.length-1];

        ModelNode deploy = getDeploy(url.toExternalForm(), name);
        ModelNode undeploy = getUndeploy(name);

        System.out.println("Deploy\n-----
\n");
        System.out.println("Formatted:\n" +
deploy.toJSONString(false));
        System.out.println("Unformatted:\n" +
deploy.toJSONString(true));
        System.out.println("\nUndeploy\n-----
-\n");
        System.out.println("Formatted:\n" +
undeploy.toJSONString(false));
        System.out.println("Unformatted:\n" +
undeploy.toJSONString(true));
    }

    public static ModelNode getUndeploy(String name)
    {
        ModelNode undeployRequest = new ModelNode();
        undeployRequest.get("operation").set("undeploy");
        undeployRequest.get("address", "deployment").set(name);

        ModelNode removeRequest = new ModelNode();
```

```

        removeRequest.get("operation").set("remove");
        removeRequest.get("address", "deployment").set(name);

        ModelNode composite = new ModelNode();
        composite.get("operation").set("composite");
        composite.get("address").setEmptyList();
        final ModelNode steps = composite.get("steps");
        steps.add(undeployRequest);
        steps.add(removeRequest);
        return composite;
    }

    public static ModelNode getDeploy(String url, String name)
    {
        ModelNode deployRequest = new ModelNode();
        deployRequest.get("operation").set("deploy");
        deployRequest.get("address", "deployment").set(name);

        ModelNode addRequest = new ModelNode();
        addRequest.get("operation").set("add");
        addRequest.get("address", "deployment").set(name);
        addRequest.get("content").get(0).get("url").set(url);

        ModelNode composite = new ModelNode();
        composite.get("operation").set("composite");
        composite.get("address").setEmptyList();
        final ModelNode steps = composite.get("steps");
        steps.add(addRequest);
        steps.add(deployRequest);
        return composite;
    }
}

```

2. 按照下列说明用命令运行这个类：

例 10.2. 执行命令

```

java -cp .:$JBOSS_HOME/modules/org/jboss/dmr/main/jboss-
dmr-1.1.1.Final-redhat-1.jar DeployDmrToJson \
file:///Users/username/support/helloWorld.war/dist/helloWorld.war

```

3. 当类运行时，下列命令格式将被显示。请根据要求使用 **deploy** 或 **undeploy** 命令。

例 10.3.

```

Deploy
-----

Formatted:
{
    "operation" : "composite",
    "address" : [],

```

```

    "steps" : [
      {
        "operation" : "add",
        "address" : {"deployment" : "helloWorld.war"},
        "content" : [{"url" :
"file:/Users/username/support/helloWorld.war/dist/helloWorld.war"}
]
      },
      {
        "operation" : "deploy",
        "address" : {"deployment" : "helloWorld.war"}
      }
    ]
  }
}
Unformatted:
{"operation" : "composite", "address" : [], "steps" :
[{"operation" : "add", "address" : {"deployment" :
"helloWorld.war"}, "content" : [{"url" :
"file:/Users/username/support/helloWorld.war/dist/helloWorld.war"}
]}, {"operation" : "deploy", "address" : {"deployment" :
"helloWorld.war"}}]}

Uneploy
-----

Formatted:
{
  "operation" : "composite",
  "address" : [],
  "steps" : [
    {
      "operation" : "undeploy",
      "address" : {"deployment" : "helloWorld.war"}
    },
    {
      "operation" : "remove",
      "address" : {"deployment" : "helloWorld.war"}
    }
  ]
}
Unformatted:
{"operation" : "composite", "address" : [], "steps" :
[{"operation" : "undeploy", "address" : {"deployment" :
"helloWorld.war"}}, {"operation" : "remove", "address" :
{"deployment" : "helloWorld.war"}}]}

```

4. 使用下列命令部署或卸载应用程序。请用上面概述的请求替换 **json request**。

例 10.4. 执行命令

```

curl -f --digest -u "<user>:<pass>" -H Content-Type:\
application/json -d '<json request>'
"http://localhost:9990/management"

```

[提交 bug 报告](#)

10.5. 用部署扫描器进行部署

10.5.1. 在部署扫描器（Deployment Scanner）里管理应用程序的部署

通过部署扫描器将应用程序部署到独立服务器实例里允许你以适合快速开发周期的方式构建和测试应用程序。你可以配置部署扫描器以适合不同应用程序类型的部署频率和行为。

[提交 bug 报告](#)

10.5.2. 用部署扫描器部署应用程序到独立服务器实例

必须具备的条件

- [第 2.1.1 节 “启动 JBoss EAP 6”](#)

概述

这个任务展示了用部署扫描器部署应用程序到独立服务器实例的方法。如 [第 10.1 节 “关于应用程序部署”](#) 所述，这个方法是为了方便开发人员而保留的，对于产品环境下的管理我们推荐管理控制台和管理 CLI 方法。

过程 10.8. 使用部署扫描器部署应用程序

1. 复制内容到 deployment 目录

复制应用程序文件到 `EAP_HOME/standalone/deployments/` 里的 deployment 目录。

2. 部署扫描模式

有两种部署方法。您可以选择自动或手动部署扫描模式。在启动部署方法之前，请先阅读 [第 10.5.8 节 “用管理 CLI 配置部署扫描器”](#)。

◦ 自动扫描

部署扫描器获取文件夹状态的变动并创建一个 [第 10.5.8 节 “用管理 CLI 配置部署扫描器”](#) 里定义的 marker 文件。

◦ 手动部署

部署扫描器需要一个 marker 文件来触发部署过程。下面的例子使用了 Unix `touch` 命令来创建一个新的 `.dodeploy` 文件。

例 10.5. 用 touch 命令进行部署

```
[user@host bin]$ touch  
$EAP_HOME/standalone/deployments/example.war.dodeploy
```

结果

应用程序文件部署到了应用服务器里。deployment 目录里创建了一个 marker 文件以表示部署成功，且应用程序在管理控制台里被标记为 **Enabled**。

例 10.6. 在部署后 deployment 目录包含了下列内容

```
example.war
example.war.deployed
```

[提交 bug 报告](#)

10.5.3. 用部署扫描器卸载独立服务器实例的应用程序

必须具备的条件

- [第 2.1.1 节 “启动 JBoss EAP 6”](#)
- [第 10.5.2 节 “用部署扫描器部署应用程序到独立服务器实例”](#)

概述

这个任务展示了用部署扫描器卸载独立服务器实例的应用程序的方法。如 [第 10.1 节 “关于应用程序部署”](#) 所述，这个方法是为了方便开发人员而保留的，对于产品环境下的管理我们推荐管理控制台和管理 CLI 方法。



注意

部署扫描器不应该和其他用于应用程序管理的部署方法一起使用。通过管理控制台从应用服务器删除的应用程序将从 runtime 删除而不会影响 marker 文件及 deployment 目录里包含的应用程序。要最小化意外重部署或其他错误的风险，在产品环境里请使用管理 CLI 和管理控制台。

过程 10.9. 用下列方法之一卸载应用程序

- **卸载应用程序**

有两种方法可以卸载应用程序，这取决于您是否想从 deployment 目录删除应用程序还是只修改它的部署状态。

- **通过删除 marker 文件进行卸载**

删除已部署的应用程序的 `example.war.deployed` marker 文件来触发部署扫描器从 runtime 卸载应用程序。

结果

部署扫描器卸载应用程序并创建一个 `example.war.undeployed` marker 文件。应用程序仍保留在 deployment 目录里。

- **通过删除应用程序进行卸载**

从 deployment 目录删除应用程序来触发部署扫描器从 runtime 卸载应用程序。

结果

部署扫描器卸载应用程序并创建一个 `filename.filetype.undeployed` marker 文件。应用程序现在不会出现在 deployment 目录里了。

结果

应用程序文件从应用服务器里卸载且不会出现在管理服务器的 **Deployments** 屏幕上了。

[提交 bug 报告](#)

10.5.4. 用部署扫描器在独立服务器实例里重新部署应用程序

必须具备的条件

- [第 2.1.1 节 “启动 JBoss EAP 6”](#)
- [第 10.5.2 节 “用部署扫描器部署应用程序到独立服务器实例”](#)

概述

这个任务展示了用部署扫描器重新部署应用程序到独立服务器实例的方法。如 [第 10.1 节 “关于应用程序部署”](#) 所述，这个方法是为了方便开发人员而保留的，对于产品环境下的管理我们推荐管理控制台和管理 CLI 方法。

过程 10.10. 在独立服务器里重新部署应用程序

- **重新部署应用程序**
重新部署用部署扫描器部署的应用程序有三种可能的方法。这些方法可以触发部署扫描器来启动部署循环，您可以按个人喜好进行选择。
 - **通过修改 marker 文件进行重新部署**
通过修改 marker 文件的访问和修改时间戳可以触发部署扫描器的重新部署。在后面的 Linux 示例里将使用 Unix **touch** 命令。

例 10.7. 用 Unix touch 命令进行重新部署

```
[user@host bin]$ touch  
EAP_HOME/standalone/deployments/example.war.dodeploy
```

结果

部署扫描器检测到 marker 文件的修改并重新部署应用程序。新的 **.deployed** 将替换旧的 marker 文件。

- **通过创建新的 .dodeploy marker 文件进行重新部署**
通过创建新的 **.dodeploy** marker 文件来触发部署扫描器的重新部署。手动部署的说明请参考 [第 10.5.2 节 “用部署扫描器部署应用程序到独立服务器实例”](#)。
- **通过删除 marker 文件进行重新部署**
如 [第 10.5.5 节 “对部署扫描器 Marker 文件的引用”](#) 所述，删除已部署应用程序的 **.deployed** marker 文件将触发卸载并创建一个 **.undeployed** marker 文件。删除卸载 marker 文件将再次触发部署循环。进一步的信息请参考 [第 10.5.3 节 “用部署扫描器卸载独立服务器实例的应用程序”](#)。

结果

应用程序被重新部署。

[提交 bug 报告](#)

10.5.5. 对部署扫描器 Marker 文件的引用

Marker 文件

Marker 文件是部署扫描器子系统的一部分。这些文件标记独立服务器的 `deployment` 目录里的应用程序的状态。Marker 文件具有和应用程序相同的名称，其后缀则表示部署的状态。下表定义了每个 marker 文件的类型及响应。

例 10.8. Marker 文件示例

下面的例子展示了用于成功部署的 `testapplication.war` 应用程序的实例的 marker 文件。

```
testapplication.war.deployed
```

表 10.1. Marker 文件类型定义

文件名后缀	来源	描述
<code>.dodeploy</code>	用户生成	表示内容应该部署到 <code>runtime</code> 或从 <code>runtime</code> 卸载。
<code>.skipdeploy</code>	用户生成	禁用应用程序的自动部署。用作禁止展开内容的自动部署的临时方法，阻止不完整的内容进入应用环境。它可以用于压缩的内容，扫描器会检测压缩内容的进度并等待完成。
<code>.isdeploying</code>	系统生成	表示部署的初始化。当部署过程完成时，Marker 文件将被删除。
<code>.deployed</code>	系统生成	表示内容已经被部署。如果文件被删除，这些内容将被卸载。
<code>.failed</code>	系统生成	表示部署失败。Marker 文件包含关于失败原因的信息。如果 Marker 文件被删除，这些内容将再次对于自动部署可见。
<code>.isundeploying</code>	系统生成	表示对删除 <code>.deployed</code> 文件的响应。完成后其内容将被卸载且 marker 文件将被自动删除。
<code>.undeployed</code>	系统生成	表示内容已被卸载。Marker 文件的删除对内容重部署没有影响。
<code>.pending</code>	系统生成	表示部署说明将被发送到有检测的问题还未解决的服务器。这个 marker 文件充当全局部署 <code>road-block</code> 。当这个条件存在时，扫描器不会指引服务器部署或卸载任何其他内容。

[提交 bug 报告](#)

10.5.6. 对部署扫描器属性的引用

部署扫描器包含下列开放给管理 CLI 且可用 `write-attribute` 进行配置的属性。关于配置选项的更多信息，请参考 [第 10.5.8 节“用管理 CLI 配置部署扫描器”](#)。

表 10.2. 部署扫描器属性

实例类型	描述	类型	默认值
auto-deploy-exploded	允许自动部署展开内容而无需 .dodeploy marker 文件。我们仅推荐用于基本的部署场景，以防止在开发人员或操作系统进行修改时发生展开的应用程序的部署。	布尔值 (Boolean)	False
auto-deploy-xml	允许自动部署 XML 内容而无需 .dodeploy marker 文件。	布尔值 (Boolean)	True
auto-deploy-zipped	允许自动部署压缩内容而无需 .dodeploy marker 文件。	布尔值 (Boolean)	True
deployment-timeout	部署扫描器在取消部署前允许尝试部署的时间。	Long	600
path	定义要扫描的实际的文件系统路径。如果指定了 relative-to 属性， path 值将充当该目录或路径的相对路径。	字符串	deployments
relative-to	对在服务器配置 XML 文件的 paths 部分定义的文件系统路径的引用。	字符串	jboss.server.base.dir
scan-enabled	允许在启动时及每隔 scan-interval 自动扫描应用程序。	布尔值 (Boolean)	True
scan-interval	扫描资料库的时间间隔（毫秒）。小于 1 的值表示扫描器只有在启动时才操作。	Int	5000

[提交 bug 报告](#)

10.5.7. 配置部署扫描器

我们可以用管理控制台或管理 CLI 配置部署扫描器。您可以创建一个新的部署扫描器或者管理现有的扫描器属性。这些包括扫描间隔、部署文件夹的位置、触发部署的应用程序文件类型。

[提交 bug 报告](#)

10.5.8. 用管理 CLI 配置部署扫描器

必须具备的条件

- [第 3.5.2 节“启动管理 CLI”](#)

概述

虽然有多种方法可以配置部署扫描器，管理 CLI 可以用批处理脚本或实时开放和修改属性。您可以用 **read-attribute** 和 **write-attribute** 全局命令行操作修改部署扫描器的行为。关于部署扫描器属性的更多信息，请参考 [第 10.5.6 节“对部署扫描器属性的引用”](#)。

部署扫描器是 JBoss EAP 6 的一个子系统，您可以在 **standalone.xml** 里查看它。

```
<subsystem xmlns="urn:jboss:domain:deployment-scanner:1.1">
  <deployment-scanner path="deployments" relative-
to="jboss.server.base.dir" scan-interval="5000"/>
</subsystem>
```

过程 10.11. 配置部署扫描器

1. 确定要配置的部署扫描器属性

通过管理 CLI 配置部署描述符要求您首先开放正确的属性名。您可以在根节点上用 **read-resources** 操作来实现，或者用 **cd** 命令来修改子节点。您也可以使用 **ls** 命令显示这个级别的属性。

◦ 用 **read-resource** 操作开放部署扫描器的属性

请使用 **read-resource** 操作来开放默认部署扫描器资源定义的属性。

```
[standalone@localhost:9999 /]/subsystem=deployment-
scanner/scanner=default:read-resource
{
  "outcome" => "success",
  "result" => {
    "auto-deploy-exploded" => false,
    "auto-deploy-xml" => true,
    "auto-deploy-zipped" => true,
    "deployment-timeout" => 600,
    "path" => "deployments",
    "relative-to" => "jboss.server.base.dir",
    "scan-enabled" => true,
    "scan-interval" => 5000
  }
}
```

◦ 用 **ls** 命令开放部署扫描器属性

请使用 **ls** 命令和 **-l** 可选参数来显示包含子系统节点、值和类型的结果。您可以输入 **ls --help** 来学习关于 **ls** 命令及其参数的更多内容。关于管理 CLI 里帮助菜单的详情，请参考 [第 3.5.5 节“用管理 CLI 获取帮助”](#)。

```
[standalone@localhost:9999 /] ls -l /subsystem=deployment-
scanner/scanner=default
```

ATTRIBUTE	VALUE	TYPE
auto-deploy-exploded	false	BOOLEAN
auto-deploy-xml	true	BOOLEAN
auto-deploy-zipped	true	BOOLEAN
deployment-timeout	600	LONG
path	deployments	STRING
relative-to	jboss.server.base.dir	STRING
scan-enabled	true	BOOLEAN
scan-interval	5000	INT

2. 用 **write-attribute** 操作配置部署扫描器

在您确定了要修改的属性的名称后，请使用 **write-attribute** 来指定属性名称和写入的新值。下面的例子都运行在子节点级别，可以通过 **cd** 命令访问，并开放默认扫描器节点的 **Tab** 完成和修改。

```
[standalone@localhost:9999 /] cd subsystem=deployment-
scanner/scanner=default
```

a. 启用展开内容的自动部署

请使用 **write-attribute** 命令来禁用展开的应用程序内容的自动部署。

```
[standalone@localhost:9999 scanner=default] :write-
attribute(name=auto-deploy-exploded,value=true)
{"outcome" => "success"}
```

b. 禁用 XML 内容的自动部署

请使用 **write-attribute** 命令来禁用 XML 应用程序内容的自动部署。

```
[standalone@localhost:9999 scanner=default] :write-
attribute(name=auto-deploy-xml,value=false)
{"outcome" => "success"}
```

c. 禁用压缩内容的自动部署

请使用 **write-attribute** 命令来禁用压缩的应用程序内容的自动部署。

```
[standalone@localhost:9999 scanner=default] :write-
attribute(name=auto-deploy-zipped,value=false)
{"outcome" => "success"}
```

d. 配置路径属性

请使用 **write-attribute** 操作来修改路径属性，用新的路径名替换 **newpathname** 以被部署扫描器监控。请注意，服务器需要重启以使修改生效。

```
[standalone@localhost:9999 scanner=default] :write-
attribute(name=path,value=newpathname)
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
```

e. 配置相对路径属性

请使用 **write-attribute** 操作来修改对 XML 配置文件里路径部分定义的文件路径的相对引用。请注意，服务器将需要重启以使修改生效。

```
[standalone@localhost:9999 scanner=default] :write-
attribute(name=relative-to,value=new.relative.dir)
{
  "outcome" => "success",
  "response-headers" => {
```

```

        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}

```

f. 禁用部署扫描器

请使用 **write-attribute** 命令并将 **scan-enabled** 设为 **false** 来禁用部署扫描器。

```

[standalone@localhost:9999 scanner=default] :write-attribute(name=scan-enabled,value=false)
{"outcome" => "success"}

```

g. 修改扫描间隔

请使用 **write-attribute** 操作来修改扫描间隔（5000 到 10000 毫秒）。

```

[standalone@localhost:9999 scanner=default] :write-attribute(name=scan-interval,value=10000)
{"outcome" => "success"}

```

结果

您对配置的修改已保存到部署扫描器里。

[提交 bug 报告](#)

10.6. 用 MAVEN 进行部署

10.6.1. 用 Maven 管理应用程序部署

通过 Maven 部署应用程序允许您将部署周期合并为现有部署工作流的一部分。

[提交 bug 报告](#)

10.6.2. 用 Maven 部署应用程序

必须具备的条件

- [第 2.1.1 节“启动 JBoss EAP 6”](#)

概述

本节展示了用 Maven 卸载应用程序的方法。下面的例子使用了 JBoss EAP 6 Quickstarts 里的 **jboss-as-helloworld.war** 应用程序。**helloworld** 项目包含了一个初始化了 **jboss-as-maven-plugin** 的 POM 文件。这个插件提供了在应用服务器里部署和卸载应用程序的简单操作。

过程 10.12. 用 Maven 部署应用程序

1. 打开终端会话并进入包含 Quickstart 例程的目录里。

例 10.9. 进入 helloworld 应用程序目录

```

[localhost]$ cd /QUICKSTART_HOME/helloworld

```

2. 运行 **Maven deploy** 命令来部署应用程序。如果应用程序已经运行，它将被重新部署。

```
[localhost]$ mvn package jboss-as:deploy
```

3. 查看结果。

- 通过在终端窗口里查看操作日志可以确认部署。

例 10.10. 通过 Maven 确认 HelloWorld 应用程序

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 32.629s
[INFO] Finished at: Fri Mar 14 09:09:50 EDT 2014
[INFO] Final Memory: 23M/204M
[INFO] -----
```

- 部署在活动应用程序服务器实例的状态流里也可以确认。

例 10.11. 通过应用服务器确认 HelloWorld 应用程序

```
09:09:49,167 INFO [org.jboss.as.repository] (management-
handler-thread - 1) JBAS014900: Content added at location
/home/username/EAP_HOME/standalone/data/content/32/4b4ef9a4bbe7
206d3674a89807203a2092fc70/content
09:09:49,175 INFO [org.jboss.as.server.deployment] (MSC
service thread 1-7) JBAS015876: Starting deployment of "jboss-
helloworld.war" (runtime-name: "jboss-helloworld.war")
09:09:49,563 INFO [org.jboss.weld.deployer] (MSC service
thread 1-8) JBAS016002: Processing weld deployment jboss-
helloworld.war
09:09:49,611 INFO [org.jboss.weld.deployer] (MSC service
thread 1-1) JBAS016005: Starting Services for CDI deployment:
jboss-helloworld.war
09:09:49,680 INFO [org.jboss.weld.Version] (MSC service thread
1-1) WELD-000900 1.1.17 (redhat)
09:09:49,705 INFO [org.jboss.weld.deployer] (MSC service
thread 1-2) JBAS016008: Starting weld service for deployment
jboss-helloworld.war
09:09:50,080 INFO [org.jboss.web] (ServerService Thread Pool -
- 55) JBAS018210: Register web context: /jboss-helloworld
09:09:50,425 INFO [org.jboss.as.server] (management-handler-
thread - 1) JBAS018559: Deployed "jboss-helloworld.war"
(runtime-name : "jboss-helloworld.war")
```

结果

应用程序已部署到应用服务器里。

[提交 bug 报告](#)

10.6.3. 用 Maven 卸载应用程序

必须具备的条件

- [第 2.1.1 节 “启动 JBoss EAP 6”](#)

概述

本节展示了用 Maven 卸载应用程序的方法。下面的例子使用了 JBoss EAP 6 Quickstarts 里的 `jboss-as-helloworld.war` 应用程序。`helloworld` 项目包含了一个初始化了 `jboss-as-maven-plugin` 的 POM 文件。这个插件提供了在应用服务器里部署和卸载应用程序的简单操作。

过程 10.13. 用 Maven 卸载应用程序

1. 打开终端会话并进入包含 Quickstart 例程的目录里。

例 10.12. 进入 helloworld 应用程序目录

```
[localhost]$ cd /QUICKSTART_HOME/helloworld
```

2. 运行 Maven `undeploy` 命令来卸载应用程序。

```
[localhost]$ mvn jboss-as:undeploy
```

3. 查看结果。

- 通过在终端窗口里查看操作日志可以确认卸载。

例 10.13. 卸载 Helloworld 应用程序的通 Maven 确认信息

```
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 1 second
[INFO] Finished at: Mon Oct 10 17:33:02 EST 2011
[INFO] Final Memory: 11M/212M
[INFO] -----
```

- 卸载在活动应用程序服务器实例的状态流里也可以确认。

例 10.14. 卸载 Helloworld 应用程序的应用服务器确认信息

```

09:51:40,512 INFO  [org.jboss.web] (ServerService Thread Pool -
- 69) JBAS018224: Unregister web context: /jboss-helloworld
09:51:40,522 INFO  [org.jboss.weld.deployer] (MSC service
thread 1-3) JBAS016009: Stopping weld service for deployment
jboss-helloworld.war
09:51:40,536 INFO  [org.jboss.as.server.deployment] (MSC
service thread 1-1) JBAS015877: Stopped deployment jboss-
helloworld.war (runtime-name: jboss-helloworld.war) in 27ms
09:51:40,621 INFO  [org.jboss.as.repository] (management-
handler-thread - 10) JBAS014901: Content removed from location
/home/username/EAP_HOME/jboss-eap-
6.3/standalone/data/content/44/e1f3c55c84b777b0fc201d69451223c0
9c9da5/content
09:51:40,621 INFO  [org.jboss.as.server] (management-handler-
thread - 10) JBAS018558: Undeployed "jboss-helloworld.war"
(runtime-name: "jboss-helloworld.war")

```

结果

从服务器服务器里卸载了应用程序。

[提交 bug 报告](#)

10.7. 控制 JBOSS EAP 6 里部署应用程序的顺序

JBoss EAP 6 提供了服务器启动时对部署顺序的细颗粒度控制。您可以启用多个 EAR 文件里的严格的部署顺序以及重启后的持久化顺序。

过程 10.14. 控制 JBoss EAP 6.0.X 里部署的顺序

1. 创建在服务器启动/停止时按顺序部署和卸载应用程序的 CLI 脚本。
2. CLI 也支持批模式，它允许您将命令和操作分组并作为一个原子单元执行。如果有一个命令或操作运行失败，该批模式里所有其他已成功执行的命令或操作将进行回滚。

过程 10.15. 控制 JBoss EAP 6.1.X 里部署的顺序

EAP 6.1.X 里有一个名为 Inter Deployment Dependencies 的新功能，它允许您在顶层部署间声明依赖关系。

1. 在 **app.ear/META-INF** 文件夹里创建一个 **jboss-all.xml** 文件（如果没有），这里的 **app.ear** 是依赖于另外一个之前部署的应用程序的归档。
2. 如下所示，在这个文件里创建一个 **jboss-deployment-dependencies** 条目。请注意，在下面的列表里，**framework.ear** 是应该在 **app.ear** 之前部署的依赖关系应用程序归档。

```

<jboss xmlns="urn:jboss:1.0">
  <jboss-deployment-dependencies xmlns="urn:jboss:deployment-
dependencies:1.0">
    <dependency name="framework.ear" />
  </jboss-deployment-dependencies>
</jboss>

```

[提交 bug 报告](#)

10.8. 部署描述符覆盖

EAP 6.1.x 里有一个新的功能，它允许您在运行时覆盖部署描述符、JAR、类、JSP 页面和其他文件。*deployment overlay* 代表了归档里的一个规则集文件，它必须被覆盖。它也提供到应该替代被覆盖文件的新文件的链接。如果在部署归档里没有出现被覆盖的文件，那它会被添加回部署里。

过程 10.16. 用管理 CLI 覆盖部署描述符

下面的步骤假设您已经有一个部署的应用程序 **app.war**，你想用 **/home/user/web.xml** 下的 **web.xml** 覆盖它的 **WEB-INF/web.xml** 文件。

1. 添加一个部署重叠并添加内容。您可以以下列两种方式进行：

- 使用 **DRM 树**

- a.

```
/deployment-overlay=myoverlay:add
```
- b.

```
/deployment-overlay=myoverlay/content=WEB-INF/web.xml:add{content={url=file:///home/user/web.xml}}
```

您可以用第二个语句添加更多的内容规则。

- 使用 **convenience 模式**

```
deployment-overlay add --name=myoverlay --content=WEB-INF/web.xml=/home/user/web.xml
```

2. 链接这个重叠至部署归档。您可以以下列两种方式进行：

- 使用 **DRM 树**

```
/deployment-overlay=myoverlay/deployment=app.war:add
```

- 使用 **convenience 模式**

```
deployment-overlay link --name=myoverlay --deployments=app.war
```

你必须指定用逗号隔开的多个归档名称。

请注意，部署归档并不需要存在于服务器上。您可以指定这个名称，但不将其链接到实际的部署。

3. 重新部署应用程序

```
/deployment=app.war:redeploy
```

[提交 bug 报告](#)

第 11 章 保护 JBOSS EAP 6

11.1. 关于安全子系统

安全 (**security**) 子系统通过 PicketBox 提供的安全服务为 Red Hat JBoss EAP 里的所有安全功能提供了基础结构。这个子系统使用了和当前请求关联的安全上下文向相关的容器开放验证管理者、授权管理者、审计管理者和映射管理者的功能。

在默认情况下，**security** 安全子系统是预配置好的，所以您很少需要修改其中的安全元素。唯一要修改的可能是是否使用 *deep-copy-subject-mode*。在多数情况下，管理员将专注于 *security domains* 的配置。

Deep Copy 模式

关于 Deep Copy 模式的细节，请参考 [第 11.4 节“关于 Deep Copy Subject 模式”](#)。

安全域

安全域 (Security Domain) 是一系列 *Java 验证和授权服务 (Java Authentication and Authorization Service, JAAS)* 的声明式安全配置，一个或多个应用程序用它来控制验证、授权、审计和映射。有三个默认的安全域：**jboss-ejb-policy**、**jboss-web-policy** 和 **other**。你也可以按照应用程序的需要创建安全域。关于安全域的细节，请参考 [第 11.6.12 节“在应用程序里使用安全域”](#)。

[提交 bug 报告](#)

11.2. 关于安全子系统的结构

安全子系统是在受管域或独立服务器的配置文件里配置的。多数配置元素可以用基于 web 的管理控制台或基于控制台的管理 CLI 来配置。下面是一个安全子系统配置的 XML 片段。

例 11.1. 安全子系统配置示例

```
<subsystem xmlns="urn:jboss:domain:security:1.2">
  <security-management>
    ...
  </security-management>
  <security-domains>
    <security-domain name="other" cache-type="default">
      <authentication>
        <login-module code="Remoting" flag="optional">
          <module-option name="password-stacking"
value="useFirstPass"/>
        </login-module>
        <login-module code="RealmUsersRoles" flag="required">
          <module-option name="usersProperties"
value="\${jboss.domain.config.dir}/application-users.properties"/>
          <module-option name="rolesProperties"
value="\${jboss.domain.config.dir}/application-roles.properties"/>
          <module-option name="realm"
value="ApplicationRealm"/>
          <module-option name="password-stacking"
value="useFirstPass"/>
        </login-module>
      </authentication>
    </security-domain>
    <security-domain name="jboss-web-policy" cache-type="default">
```



```

        <authorization>
            <policy-module code="Delegating" flag="required"/>
        </authorization>
    </security-domain>
    <security-domain name="jboss-ejb-policy" cache-type="default">
        <authorization>
            <policy-module code="Delegating" flag="required"/>
        </authorization>
    </security-domain>
</security-domains>
<vault>
    ...
</vault>
</subsystem>

```

<security-management>、**<subject-factory>** 和 **<security-properties>** 元素没有出现在默认配置里。从 JBoss EAP 6.1 开始已启用了 **<subject-factory>** 和 **<security-properties>** 元素。

[提交 bug 报告](#)

11.3. 配置安全子系统

你可以用管理 CLI 或基于 web 的管理控制台来配置安全子系统。

安全子系统里的每个顶级元素都包含关于安全配置不同方面的信息。关于安全子系统配置的例子，请参考第 11.2 节“关于安全子系统的结构”。

<security-management>

这部分内容覆盖了安全子系统的高层行为。每个设置都是可选的。除了 Deep Copy 模式，须该这些设置的任何一个都是不寻常的。

选项	描述
deep-copy-subject-mode	指定是否复制或链接安全令牌以用于额外的线程安全。
authentication-manager-class-name	指定一个要使用的其他的 AuthenticationManager 实现的类名。
authorization-manager-class-name	指定一个要使用的其他的 AuthorizationManager 实现的类名。
audit-manager-class-name	指定一个要使用的其他的 AuditManager 实现的类名。
identity-trust-manager-class-name	指定一个要使用的其他的 IdentityTrustManager 实现的类名。
mapping-manager-class-name	指定一个要使用的 MappingManager 实现的类名。

<subject-factory>

主题工厂（Subject factory）控制主题实例的创建。它可以使用验证管理者来检验调用者。主题工厂的主要用途是为了 JCA 组件建立主题。你通常不需要修改它。

<security-domains>

保存多个安全域的容器元素。安全域可能包含关于验证、授权、映射、审计模块以及 JASPI 验证和 JSSE 配置的信息。你的应用程序可以指定一个安全域来管理它的安全信息。

<security-properties>

包含在 `java.security.Security` 类上设置的属性的名字和值。

[提交 bug 报告](#)

11.4. 关于 DEEP COPY SUBJECT 模式

如果 *Deep Copy 主题模式* 被禁用（默认），复制安全数据结构会产生一个对原始结构的引用，而不是复制整个数据结构。这个行为效率更高，但在具有相同标识符的多个线程通过冲刷或登出操作清除主题时容易受数据损坏的影响。

只要被标记为可克隆的（cloneable），Deep Copy 主题模式会导致数据结构及相关数据的完整复制。这是更多线程安全的，但效率更低。

Deep Copy Subject 模式是作为安全子系统的一部分来配置的。

[提交 bug 报告](#)

11.5. 启用 DEEP COPY SUBJECT 模式

你可以通过基于 Web 的管理控制台或管理 CLI 来启用 Deep Copy 安全模式。

过程 11.1. 通过管理控制台启用 Deep Copy 安全模式

1. 登录到管理控制台。

请参考 [第 3.4.2 节“登录到管理控制台”](#)。

2. 受管域：选择合适的配置集。

在受管域里，安全子系统是针对每个配置集进行配置的，而且你可以在每个配置集里独立地启用或禁用 Deep Copy 安全模式。

要选择配置集，请点击控制台顶部的 **Configuration** 标签，然后在左上角的 **Profile** 下拉菜单里选择配置集。

3. 打开 Security Subsystem 配置菜单。

展开 **Security** 菜单，然后选择 **Security Subsystem**。

4. 启用 Deep Copy Subject 模式。

点击 **Edit** 按钮。选定 **Deep Copy Subjects** 复选框来启用 Deep Copy Subject 模式。

使用管理 CLI 启用 Deep Copy Subject 模式

如果你想通过管理 CLI 来启用这个选项，请使用下列命令。

例 11.2. 受管域

```
/profile=full/subsystem=security/:write-attribute(name=deep-copy-
subject-mode,value=TRUE)
```

例 11.3. 独立服务器

```
/subsystem=security/:write-attribute(name=deep-copy-subject-
mode,value=TRUE)
```

[提交 bug 报告](#)

11.6. 安全域

11.6.1. 关于安全域

安全域是 JBoss EAP 6 安全子系统的一部分。所有的安全配置现在都由受管域的域控制器或独立服务器来集中管理了。

安全域由验证、授权、安全映射和审计的配置组成。它实现了 *Java 验证和授权服务 (Java Authentication and Authorization Service, JAAS)* 声明式安全性。

验证指的是检验用户的身份。按照安全性术语，这个用户被称为 *principal*。虽然验证的授权是不同的，系统包含的许多验证模块也可以处理授权。

授权 (Authorization) 是一个服务器确定已验证的用户是否具有访问系统或操作里某些权利或资源的权限的过程。

安全映射 (Security mapping) 指的是将信息传递给应用程序之前在 *principal*、角色或属性里添加、修改、删除信息的能力。。

审计管理者允许你配置 *提供者模块 (provider modules)* 来控制报告安全事件的方式。

如果你使用安全域，你可以从应用程序删除所有的专有安全配置。这允许你集中地修改安全参数。受益于这种配置结构的一个常见场景是在测试和产品环境间移动应用程序。

[提交 bug 报告](#)

11.6.2. 关于 Picketbox

Picketbox 是基础的安全框架，它为运行在 JBoss EAP 6 里的 Java 应用程序提供了验证、授权、审计和映射能力。它用单一的配置在单一的框架里提供了这些能力：

- [第 11.6.3 节 “关于验证”](#)
- [第 11.6.5 节 “关于授权”](#) 和访问控制
- [第 11.6.7 节 “关于安全性审计”](#)
- [第 11.6.10 节 “关于安全性映射”](#) 和 *principal*、角色、属性

[提交 bug 报告](#)

11.6.3. 关于验证

验证指的是确定一个主题并检验标识的真实性。最常见的验证机制是用户名和密码的组合。其他常见的机制使用共享密钥、智能卡或指纹。按照 Java EE 的声明安全性，成功验证的结果被称为 **principal**。

JBoss EAP 6 使用一个可插拔的验证模块系统，它提供灵活性以及与机构里已使用的验证系统的集成性。每个安全域都包含一个或多个配置好的验证模块。每个模块都包含其他的配置参数来自定义其行为。配置验证子系统的最简单的方法是通过基于 **web** 的管理控制台。

验证和授权不同，虽然他们经常联系在一起。许多包含的验证模块也可以处理授权。

[提交 bug 报告](#)

11.6.4. 配置安全域的验证

要配置安全域的验证，请登录到管理控制台并遵循下列步骤。

过程 11.2. 为安全域设置验证

1. 打开安全域的详细视图。

- a. 点击管理控制台顶部的 **Configuration** 标签。
- b. 从 **Profile** 视图左上角的 **Profile** 里选择要修改的配置集。
- c. 展开 **Security** 菜单，然后选择 **Security Domains**。
- d. 点击您要编辑的安全域的 **View** 链接。

2. 进入验证子系统配置。

如果还未选择的话，选择视图顶部的 **Authentication** 标签。

配置区域分成两部分：**Login Modules** 和 **Details**。登录模块是配置的基本单元。安全域可以包含多个登录模块，每个都包括几个属性和选项。

3. 添加一个验证模块。

点击 **Add** 添加一个 JAAS 验证模块。填写模块的细节。

Code 是模块的类名。**Flag** 设置控制模块如何和相同安全域里的其他验证模块关联。

对标签的解释

Java EE 6 规格提供了安全域的标签的解释。下面的列表来自

<http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html#Ap>
关于更消息的信息，请参考这个文档。

标签	详情
required	登录模块是验证成功所必需的。如果成功或失败，验证都仍会继续处理登录模块列表。
requisite	登录模块是验证成功所必需的。如果成功，验证将继续处理登录模块列表。如果失败，控制权马上返回给应用程序（验证不会继续处理登录模块列表）。

标签	详情
sufficient	登录模块不是验证成功所必需的。如果成功，控制权马上返回给应用程序（验证不会继续处理登录模块列表）。如果失败，验证将继续处理登录模块列表。
optional	登录模块不是验证成功所必需的。如果成功或失败，验证都仍会继续处理登录模块列表。

4. 编辑验证设置

在你添加了模块时，你可以通过屏幕上的 **Details** 里的 **Edit** 按钮修改它的 **Code** 或 **Flags**。请确保选择了 **Attributes** 标签页。

5. 可选的：添加或删除模块选项。

如果你需要在模块里添加选项，请点击 **Login Modules** 列表里的条目，并在 **Details** 页面里选择 **Module Options** 标签页。点击 **Add** 按钮，并提供这个选项的键和值。你可以用 **Remove** 按钮来删除选项。

结果

你的验证模块已添加至安全域，且马上可为使用安全域的应用程序所用。

jboss.security.security_domain 模块选项

在默认情况下，安全域里定义的每个登录模块都会自动添加一个 **jboss.security.security_domain** 模块选项。这个选项会给检查是否只定义了已知选项的登录模块带来问题。IBM Kerberos 登录模块 **com.ibm.security.auth.module.Krb5LoginModule** 就是其中之一。

你可以通过在启动 JBoss EAP 时设置系统属性为 **true** 来禁用添加这个模块选项的行为。请添加下列内容到你的启动参数里。

```
-Djboss.security.disable.secdomain.option=true
```

你也可以用基于 web 的管理控制台来设置这个属性。在独立服务器里，你可以在配置文件的 **Profile** 部分来设置系统属性。在受管域里，你可以对每个服务器组设置系统属性。

[提交 bug 报告](#)

11.6.5. 关于授权

授权是一个基于身份来赋予或拒绝对资源访问的机制。它作为一系列可以赋予 **principal** 的声明式安全角色来实现。

JBoss EAP 使用一个模块化系统来配置授权。每个安全域都可以获得一个或多个授权策略。每个策略都有一个定义自身行为的基本模块。它也可以通过特定的标记和属性来配置。配置授权子系统的最简单的方法是使用基于 web 的管理控制台。

授权不同于验证，它通常在验证之后发生。很多验证模块也可以处理授权。

[提交 bug 报告](#)

11.6.6. 配置安全域里的授权

要配置安全域的授权，请登录到管理控制台并遵循下列步骤。

过程 11.3. 在安全域里设置授权

- 1. 打开安全域的详细视图。
 - a. 点击管理控制台顶部的 **Configuration** 标签。
 - b. 在受管域里，从左上角的 **Profile** 下拉菜单里选择要修改的配置集。
 - c. 展开 **Security** 菜单，然后选择 **Security Domains**。
 - d. 点击您要编辑的安全域的 **View** 链接。

- 2. 进入授权子系统配置。
从屏幕顶部选择 **Authorization** 标签。

配置区域分成两部分：**Policies** 和 **Details**。登录模块是配置的基本单元。安全域可以包含几个授权策略，每个都包括几个属性和选项。

- 3. 添加策略
点击 **Add** 添加一个 JAAS 授权策略模块。填写模块的细节。

Code 是模块的类名。**Flag** 控制模块如何和相同安全域里的其他授权策略模块关联。

对标签的解释

Java EE 6 规格提供了安全域的标签的解释。下面的列表来自 <http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html#Ap> 关于更消息的信息，请参考这个文档。

标签	详情
required	登录模块是授权成功所必需的。如果成功或失败，授权都仍会继续处理登录模块列表。
requisite	登录模块是授权成功所必需的。如果成功，授权将继续处理登录模块列表。如果失败，控制权马上返回给应用程序（授权不会继续处理登录模块列表）。
sufficient	登录模块不是授权成功所必需的。如果成功，控制权马上返回给应用程序（授权不会继续处理登录模块列表）。如果失败，授权将继续处理登录模块列表。
optional	登录模块不是授权成功所必需的。如果成功或失败，授权都仍会继续处理登录模块列表。

- 4. 编辑授权设置
在你添加了模块时，你可以通过屏幕上的 **Details** 里的 **Edit** 按钮修改它的 **Code** 或 **Flags**。请确保选择了 **Attributes** 标签页。
- 5. 可选的：添加或删除模块选项。

如果你需要在模块里添加选项，请点击 **Policies** 列表里的条目，并在 **Details** 页面里选择 **Module Options** 标签页。点击 **Add** 按钮，并提供这个选项的键和值。你可以用 **Remove** 按钮来删除选项。

结果

你的授权模块已添加至安全域，且马上可为使用安全域的应用程序所用。

[提交 bug 报告](#)

11.6.7. 关于安全性审计

安全性审计包括触发事件，如在响应安全子系统里发生的事件而写入日志。审计机制和验证、授权、安全性映射细节都是作为安全域的一部分来配置的。

审计使用*提供者模块*。你可以使用内含的模块，或者实现自己的模块。

[提交 bug 报告](#)

11.6.8. 配置安全审计

要配置安全域的安全审计，请登录到管理控制台并遵循下列步骤。

过程 11.4. 在安全域里设置安全审计

1. 打开安全域的详细视图。
 - a. 点击屏幕顶部的 **Configuration**。
 - b. 在受管域里，从左上角的 **Profile** 选择框里选择要修改的配置集。
 - c. 展开 **Security** 菜单，然后选择 **Security Domains**。
 - d. 点击您要编辑的安全域的 **View** 链接。
2. 进入审计子系统配置。
从屏幕顶部选择 **Audit** 标签页。

配置区域分成两部分：**Provider Modules** 和 **Details**。提供者模块 (Provider Module) 是配置的基本单元。安全域可以包含多个提供者模块，每个都包括几个属性和选项。

3. 添加一个提供者模块。
点击 **Add**。在 **Code** 里输入提供者模块的类名。

4. 检验你的模块是否可以运行

审计模块的目的是提供一个在安全子系统里监控事件的途径。这种监控可以通过写入日志文件、电子邮件通知或其他可度量的审计机制来实现。

例如，JBoss EAP 6 默认包含了 **LogAuditProvider** 模块。如果按照上面的步骤启用，这个审计模块会将安全通知写入 **EAP_HOME** 目录里的 **log** 子目录下的 **audit.log** 文件里。

要检验上面的步骤是否可以在 **LogAuditProvider** 上下文里运行，你可以执行一个可能触发通知的动作并检查审计日志文件。

关于安全审计提供者模块的完整列表，请参考 [第 12.4 节 “包括的安全审计供应商模块”](#)。

5. 可选的：添加、编辑或删除模块选项。

如果你需要在模块里添加选项，请点击 **Modules** 列表里的条目，并在 **Details** 页面里选择 **Module Options** 标签页。点击 **Add** 按钮，并提供这个选项的键和值。

要编辑已存在的选项，请点击 **Remove** 删除它，然后点击 **Add** 再次以正确的选项添加它。

结果

你的安全审计模块已添加至安全域，且马上可为使用安全域的应用程序所用。

[提交 bug 报告](#)

11.6.9. 关于审计日志

如果启用了 **LogAuditProvider** 模块，JBoss EAP 6 将维护一个审计日志，它记录应用程序和登录模块里的验证和授权。这个文件默认名为 **audit.log** 且位于 **EAP_HOME** 的 **log** 子目录里。这个行为是由日志子系统的日志处理程序确定的。此外，**LogAuditProvider** 模块的输出可以通过 **syslog** 日志处理程序发送到 **syslog** 服务器而不是某个文件。

在默认情况下，**LogAuditProvider** 模块只输出到累积的 **audit.log** 文件里。要实现定期的轮询文件处理程序 **AUDIT**，请使用下列管理 CLI 命令。

```
/subsystem=logging/periodic-rotating-file-handler=AUDIT/:add(suffix=.yyyy-MM-dd,formatter=%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n,level=TRACE,file={"relative-to" => "jboss.server.log.dir","path" => "audit.log"})
```

还可查看：

- [第 14.1.12 节“关于日志处理程序”](#)

[提交 bug 报告](#)

11.6.10. 关于安全性映射

安全性映射允许你在验证或授权发生后但在信息传递给应用程序之前合并验证和授权信息。其中一个例子是使用用于验证的 **X509** 证书，然后从证书转换 **principal** 为你的应用程序可以显示的逻辑名称。

你可以映射 **principal**（验证）、角色（授权）或凭证（非 **principal** 或角色的属性）。

角色映射用于在验证后对主题添加、替换或删除角色。

Principal 映射用于在验证后修改 **principal**。

属性映射用来从外部系统转换属性以供应用程序使用，反之亦然。

[提交 bug 报告](#)

11.6.11. 在安全域里配置安全映射

要配置安全域的安全映射，请登录到管理控制台并遵循下列步骤。

过程 11.5. 在安全域里设置安全映射

1. 打开安全域的详细视图。

- a. 点击管理控制台顶部的 **Configuration** 标签。
- b. 在受管域里，从左上角的 **Profile** 选择框里选择配置集。
- c. 展开 **Security** 菜单，然后选择 **Security Domains**。
- d. 点击您要编辑的安全域的 **View** 链接。

2. 进入映射子系统配置。

从屏幕顶部选择 **Mapping** 标签。

配置区域分成两部分：**Modules** 和 **Details**。映射模块是配置的基本单元。安全域可以包含多个映射模块，每个都包括几个属性和选项。

3. 添加安全映射模块。

点击 **Add**。

输入模块的相关内容。**Code** 是模块的类名。**Type** 字段表示这个模块执行的映射的类型。所允许的值有 **principal**、**role**、**attribute** 和 **credential**。

4. 编辑安全映射模块

在你添加了模块时，你可以修改它的 **Code** 或 **Type**。

- a. 选择 **Attributes** 标签页。
- b. 点击屏幕的 **Details** 部分里的 **Edit** 按钮。

5. 可选的：添加、编辑或删除模块选项。

如果你需要在模块里添加选项，请点击 **Modules** 列表里的条目，并在 **Details** 页面里选择 **Module Options** 标签页。点击 **Add** 按钮，并提供这个选项的键和值。

要编辑已存在的选项，请先点击 **Remove** 删除它，然后用新的值再次添加它。

使用 **Remove** 按钮来删除选项。

结果

你的安全映射模块已添加至安全域，且马上可为使用安全域的应用程序所用。

[提交 bug 报告](#)

11.6.12. 在应用程序里使用安全域

介绍

要在应用程序里使用安全域，首先您必须在服务器配置文件里定义安全域并在应用程序的描述符文件里启用安全域。然后您必须添加所需的注解到使用安全域的 EJB。本节涵盖了在应用程序里使用安全域所需的步骤。



警告

如果应用程序不是使用验证缓存的安全域的一部分，用于该应用程序的用户验证将也可用于安全域里的其他应用程序。

过程 11.6. 配置你的应用程序以使用安全域

1. 定义安全域

您需要在服务器的配置文件里定义安全域，然后在应用程序的描述符文件里启用它。

a. 在服务器的配置文件里配置安全域

安全域是在服务器配置文件的 **security** 子系统里配置的。如果 JBoss EAP 6 实例运行在受管域里，配置文件应该是 **domain/configuration/domain.xml**。如果是独立服务器，则是 **standalone/configuration/standalone.xml** 文件。

other、**jboss-web-policy** 和 **jboss-ejb-policy** 都是 JBoss EAP 6 里默认提供的安全域。下面的 XML 示例是从服务器配置文件的 **security** 子系统里复制的。

为执行更快的验证检查，安全域的 **cache-type** 属性指定了一个缓存。允许的值由用于简单表缓存的 **default** 或 Infinispan 缓存的 **infinispan**。

```
<subsystem xmlns="urn:jboss:domain:security:1.2">
  <security-domains>
    <security-domain name="other" cache-type="default">
      <authentication>
        <login-module code="Remoting" flag="optional">
          <module-option name="password-stacking"
value="useFirstPass"/>
        </login-module>
        <login-module code="RealmDirect"
flag="required">
          <module-option name="password-stacking"
value="useFirstPass"/>
        </login-module>
      </authentication>
    </security-domain>
    <security-domain name="jboss-web-policy" cache-
type="default">
      <authorization>
        <policy-module code="Delegating"
flag="required"/>
      </authorization>
    </security-domain>
    <security-domain name="jboss-ejb-policy" cache-
type="default">
      <authorization>
        <policy-module code="Delegating"
flag="required"/>
      </authorization>
    </security-domain>
  </security-domains>
</subsystem>
```

```

        </security-domain>
    </security-domains>
</subsystem>

```

你可以按需要用管理控制台或 CLI 配置其他的安全域。

b. 在应用程序的描述符文件里启用安全域

安全域是在应用程序的 **WEB-INF/jboss-web.xml** 文件里的 **<jboss-web>** 元素的 **<security-domain>** 子元素里指定的。下面的例子配置了一个名为 **my-domain** 的安全域。

```

<jboss-web>
    <security-domain>my-domain</security-domain>
</jboss-web>

```

这只是你可以在 **WEB-INF/jboss-web.xml** 描述符里指定的许多设置中的一个。

2. 在 EJB 里添加必需的注解

你可以用 **@SecurityDomain** 和 **@RolesAllowed** 注解在 EJB 里配置安全性。下面的 EJB 代码示例限制了具有 **guest** 角色的用户对 **other** 安全域的访问。

```

package example.ejb3;

import java.security.Principal;

import javax.annotation.Resource;
import javax.annotation.security.RolesAllowed;
import javax.ejb.SessionContext;
import javax.ejb.Stateless;

import org.jboss.ejb3.annotation.SecurityDomain;

/**
 * Simple secured EJB using EJB security annotations
 * Allow access to "other" security domain by users in a "guest"
 * role.
 */
@Stateless
@RolesAllowed({ "guest" })
@SecurityDomain("other")
public class SecuredEJB {

    // Inject the Session Context
    @Resource
    private SessionContext ctx;

    /**
     * Secured EJB method using security annotations
     */
    public String getSecurityInfo() {
        // Session context injected using the resource annotation
        Principal principal = ctx.getCallerPrincipal();
        return principal.toString();
    }
}

```

关于更多的代码示例，请参考 JBoss EAP 6 Quickstarts 集里的 **ejb-security quickstart**，你可以在红帽的客户门户找到这些例子。

[提交 bug 报告](#)

11.6.13. Java 容器授权合约 (JACC)

11.6.13.1. 关于 Java 容器授权合约 (JACC)

Java 容器授权合约 (Java Authorization Contract for Containers, JACC) 是一个定义容器和授权服务提供商之间合约的标准，它规范容器使用的提供者实现。它是由 JSR-115 定义的，你可以在 [Java Community Process](http://jcp.org/en/jsr/detail?id=115) 网站上找到相关信息：<http://jcp.org/en/jsr/detail?id=115>。从 Java EE 1.3 以后，它已经是核心 Java EE 规格的一部分了。

JBoss EAP 通过安全子系统的安全性功能实现了对 JACC 的支持。

[提交 bug 报告](#)

11.6.13.2. 配置 Java 容器授权合约 (JACC) 的安全性

要配置 JACC，你需要用正确的模块配置你的安全域，然后修改 **jboss-web.xml** 来包含正确的参数。

为安全域添加 JACC 支持

要为安全域添加 JACC 支持，请添加 **JACC** 授权策略到安全域的授权栈里，并设置 **required** 标记。下面是一个带有 JACC 支持的安全域的例子。然而，安全域是在管理控制台或 CLI 里，而不是直接在 XML 里配置的。

```
<security-domain name="jacc" cache-type="default">
  <authentication>
    <login-module code="UsersRoles" flag="required">
    </login-module>
  </authentication>
  <authorization>
    <policy-module code="JACC" flag="required"/>
  </authorization>
</security-domain>
```

配置 Web 应用程序以使用 JACC

jboss-web.xml 位于你的部署的 **META-INF/** 或 **WEB-INF/** 目录里，且包含用 web 容器的覆盖选项和其他的 JBoss 专有的配置。要使用启用了 JACC 的安全域，你需要包括 **<security-domain>** 元素并设置 **<use-jboss-authorization>** 元素为 **true**。下面的应用程序是用上面的 JACC 安全域进行正确配置的。

```
<jboss-web>
  <security-domain>jacc</security-domain>
  <use-jboss-authorization>true</use-jboss-authorization>
</jboss-web>
```

配置 EJB 应用程序来使用 JACC

配置 EJB 使用安全域并使用 JACC 对于不同的 Web 应用程序是不同的。对于 EJB，你可以在 **ejb-**

`jar.xml` 里为一个方法或方法组声明 *method permissions*。在 `<ejb-jar>` 元素里，任何子 `<method-permission>` 元素都包含关于 JACC 角色的信息。详情请参考示例配置。`EJBMethodPermission` 类是 Java EE 6 API 的一部分，且 <http://docs.oracle.com/javaee/6/api/javax/security/jacc/EJBMethodPermission.html> 里有相关的文档。

例 11.4. EJB 里的 JACC 方法权限示例

```
<ejb-jar>
  <assembly-descriptor>
    <method-permission>
      <description>The employee and temp-employee roles may access any
method of the EmployeeService bean </description>
      <role-name>employee</role-name>
      <role-name>temp-employee</role-name>
      <method>
        <ejb-name>EmployeeService</ejb-name>
        <method-name>*</method-name>
      </method>
    </method-permission>
  </assembly-descriptor>
</ejb-jar>
```

你也可以通过安全域约束 EJB 的验证和授权机制，就像对 Web 应用程序所做的一样。安全域在 `jboss-ejb3.xml` 描述符里的 `<security>` 子元素里声明。除了安全域以外，你也可以指定 *run-as principal*，它可以修改运行 EJB 的 *principal*。

例 11.5. EJB 里的安全域声明示例

```
<security>
  <ejb-name>*</ejb-name>
  <security-domain>myDomain</security-domain>
  <run-as-principal>myPrincipal</run-as-principal>
</security>
```

[提交 bug 报告](#)

11.6.14. Java 容器验证 SPI (JASPI)

11.6.14.1. 关于 Java 容器验证 SPI (JASPI) 的安全性

容器安全性的 Java 验证 SPI (JASPI 或 JASPIC) 是一个 Java 程序的可插拔接口。它是在 Java Community Process 的 JSR-196 里定义的。关于它的规格详情，请参考 <http://www.jcp.org/en/jsr/detail?id=196>。

[提交 bug 报告](#)

11.6.14.2. 配置 Java 容器验证 SPI (JASPI) 的安全性

要验证 JASPI 提供者，请添加 `<authentication-jaspi>` 元素到你的安全域里。其配置和标准的验证模块类似，但登录模块元素包含在 `<login-module-stack>` 元素里。它的配置的结构是：

例 11.6. authentication-jaspi 元素的结构

```
<authentication-jaspi>
  <login-module-stack name="...">
    <login-module code="..." flag="...">
      <module-option name="..." value="..." />
    </login-module>
  </login-module-stack>
  <auth-module code="..." login-module-stack-ref="...">
    <module-option name="..." value="..." />
  </auth-module>
</authentication-jaspi>
```

登录模块自身也以标准验证模块完全相同的方式进行配置。

因为基于 web 的管理控制台没有开放 JASPI 验证模块的配置，在直接添加配置到 `EAP_HOME/domain/configuration/domain.xml` 或 `EAP_HOME/standalone/configuration/standalone.xml` 之前你需要完全停止 JBoss EAP。

[提交 bug 报告](#)

11.7. 保证 IIOP 的安全

11.7.1. 关于 JBoss IIOP

IIOP (Internet Inter-ORB Protocol) 是 CORBA 客户用来调用 CORBA 服务器上的远程功能的通讯协议。如 EJB 规格所定义的，JBoss IIOP 支持 CORBA/IIOP 对部署在 JBoss EAP 服务器上的 Enterprise Bean 的访问。它使 Enterprise Bean 的方法可用于用 Java 编写的 RMI 和 IIOP 客户或用 Java、C++ 或其他语言编写的 CORBA 客户。IIOP 引擎可用其他完全兼容 CORBA 的引擎替换。默认的引擎是 JacORB，它是 CORBA 标准的一个免费实现。

[提交 bug 报告](#)

11.7.2. 关于 IOR

CORBA 系统里使用一个互用的对象引用 (Interoperable Object Reference, IOR) 来标识对象。它由下列元素组成：

- 对象类型。
- 服务器的主机名。
- 服务器的端口号。
- 标识对象的键。

主机名和端口号用来与服务器通讯，而服务器使用对象键来标识对象。

[提交 bug 报告](#)

11.7.3. IOR 安全参数

必须具备的条件

- 用下列管理 CLI 命令启用 IOR 设置。

```
/subsystem=jacorb/ior-settings=default:add
```

- 确保已启用 JacORB 子系统。例如，它已在 **full** 配置集里启用，但没有在 **default (web)** 配置集里启用。关于如何启用 JacORB 子系统的细节，请参考 [第 21.4.2 节“为 JTS 事务配置 ORB”](#)。

`iorSASContextType` 指定用于设置 IOR 安全属性服务 (IOR Secure Attribute Service) 的属性。这些参数只能用管理 CLI 进行设置。

表 11.1. `iorSASContextType`

参数	描述	有效值
caller-propagation	指定是否应该在 SAS 上下文里传播调用者	none、supported

```
/subsystem=jacorb/ior-settings=default/setting=sas-context:add
/subsystem=jacorb/ior-settings=default/setting=sas-context:write-attribute(name=caller-propagation, value=NONE|SUPPORTED)
```

`iorASContextType` 指定用来设置 IOR 验证服务 (Authentication Service) 的属性。下列属性都是可选的。

表 11.2. `iorASContextType`

参数	描述	类型	有效值
auth-method	验证方法。	字符串	none、username_password
realm	验证服务的区名。	字符串	默认值： Default
required	指定是否要求验证。	布尔值 (Boolean)	true、false

```
/subsystem=jacorb/ior-settings=default/setting=as-context:add
/subsystem=jacorb/ior-settings=default/setting=as-context:write-attribute(name=ATTRIBUTE, value=VALUE)
```

`iorTransportconfigType` 指定用于 IOR 传输设置的属性。

表 11.3. `iorTransportconfigType`

参数	描述	有效值
integrity	指定传输是否要求完整性保护。	none 、 supported 或 required 。
confidentiality	指定传输是否要求保密性保护。	none 、 supported 或 required 。
trust-in-target	指定传输是否必须信息要求建立的目标。	none 、 supported
trust-in-client	指定传输是否必须信息要求建立的客户。	none 、 supported 或 required 。
detect-replay	指定传输是否要求重放检测 (Replay Detection) 。	none 、 supported 或 required 。
detect-misordering	指定传输是否要求错序检测。	none 、 supported 或 required 。

```
/subsystem=jacorb/ior-settings=default/setting=transport-config:add
/subsystem=jacorb/ior-settings=default/setting=transport-config:write-attribute(name=ATTRIBUTE, value=VALUE)
```

[提交 bug 报告](#)

11.8. 管理接口的安全性

11.8.1. 默认的用户安全性配置

介绍

在 EAP 6 里，所有的管理接口都默认是有设置安全性的。这个安全性采取两种形式：

- 本地接口通过本地客户和服务端间的 **SASL** 合约设置安全性。这个安全机制基于客户访问本地文件系统的能力。这是因为访问本地文件系统会允许客户添加用户或修改配置以阻挠其他安全机制。如果对文件系统的物理访问可以实现，那么其他安全机制就是多余的。这个机制以四个步骤实现：



注意

即使你通过 HTTP 连接本地主机，HTTP 访问仍会视作远程的。

1. 客户发送一条消息给服务器，它包含一个用本地 **SASL** 机制验证的请求。
2. 服务器生成一个一次性的令牌，将其写入到唯一的文件里，然后发送具有完整文件路径的消息给客户。
3. 客户从文件里读取令牌并发送给服务器，检验它是否具有对文件系统的本地访问权限。
4. 服务器验证令牌并删除这个文件。

- 远程客户，包括本地 HTTP 客户，使用基于区的安全性。带有使用管理接口远程配置 JBoss EAP 6 权限的默认区是 **ManagementRealm**。我们提供了一个脚本，允许你添加用户到这个区（或者你创建的区）。请参考《JBoss EAP 6 安装指南》的 *Getting Started* 章节。对于每个用户、用户名、hashed 密码、以及区都存储在文件里。

受管域

EAP_HOME/domain/configuration/mgmt-users.properties

独立服务器

EAP_HOME/standalone/configuration/mgmt-users.properties

即使 **mgmt-users.properties** 的内容都是以掩码显示的，这个文件仍必须作为敏感文件对待。我们推荐将其文件权限设置为 **600**，这样除了文件所有者，其他人没有读或写的权限。

提交 bug 报告

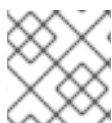
11.8.2. 高级管理接口配置概述

EAP_HOME/domain/configuration/host.xml 或

EAP_HOME/standalone/configuration/standalone.xml 里的管理接口配置控制主机控制器进程绑定哪些网络接口，哪种管理接口是可用的，哪种类型的验证系统用来验证每个接口上的用户。本主题讨论了如何配置管理接口以适应你的运行环境。

管理子系统由包含几个可配置属性以及下面三个可配置子元素的 **<management>** 元素组成。首先定义安全区和转出连接，然后再将它们作为属性应用到管理接口。

- **<security-realms>**
- **<outbound-connections>**
- **<management-interfaces>**
- **<audit-log>**



注意

请参阅《管理和配置指南》里的『*Management Interface Audit Logging*』章节。

安全区

安全区（Security Realm）负责允许通过管理 API、管理 CLI 和基于 web 的管理控制台管理 JBoss EAP 6 的用户的验证和授权。

默认安装里有两种不同的基于文件的安全区：**ManagementRealm** 和 **ApplicationRealm**。每个安全区都使用一个 **-users.properties** 文件来存储用户和哈希密码，以及一个 **-roles.properties** 来存储用户和角色间的映射。相同也包含了对启用了 LDAP 的安全区的支持。



注意

安全区也可以用于你自己的应用程序。这里讨论的安全区是管理接口所专有的。

转出的连接

一些安全区连接至外部接口，如 LDAP 服务器。转出连接（Outbound connection）定义了如何创建这种连接。预定义的连接类型，**ldap-connection**，设置了连接 LDAP 服务器并验证凭证的所有必需和可选的属性。

关于配置 LDAP 验证的更多信息，请参考 [第 11.8.4 节“在管理接口里使用 LDAP 进行验证”](#)。

管理接口

管理接口包含如何连接和配置 JBoss EAP 的属性。这样的信息包括命名网络接口、端口、安全区和其他关于接口的可配置信息。默认安装里包含了两个接口：

- **http-interface** 是基于 web 的管理控制台的配置。
- **native-interface** 是命令行管理 CLI 和类 REST 管理 API 的配置。

主机管理子系统的每个主要可配置元素都是相关的。安全区引用转出连接，而管理接口引用安全区。

相关的信息请参考：[第 11.10.1 节“保护管理接口”](#)。

[提交 bug 报告](#)

11.8.3. 关于 LDAP

轻量级目录访问协议（Lightweight Directory Access Protocol，LDAP）是一个在网络里存储和分发目录信息的协议。这个目录信息包含关于用户、硬件设备、访问角色和限制以及其他信息。

LDAP 的一些常见实现包括 OpenLDAP、Microsoft Active Directory、IBM Tivoli Directory Server、Oracle Internet Directory 等。

JBoss EAP 6 带有几个验证和授权模块，它们允许在 Web 和 EJB 应用程序里将 LDAP 服务器用于验证和授权。

[提交 bug 报告](#)

11.8.4. 在管理接口里使用 LDAP 进行验证

要在管理控制台、管理 CLI 或 API 里将 LDAP 目录服务器用作验证源，你需要执行下列过程:

1. 创建一个到 LDAP 服务器的转出连接。
2. 创建一个启用 LDAP 的安全区。
3. 在管理接口里引用新的安全区。

创建一个到 LDAP 服务器的转出连接

LDAP 转出连接允许下列属性：

表 11.4. LDAP 转出连接的属性

属性	Required	描述
url	是	目录服务器的 URL 地址。

属性	Required	描述
search-dn	否	授权执行搜索的用户的全限定可区分名称 (Distinguished Name , DN) 。
search-credentials	否	用户授权执行搜索的密码。
initial-context-factory	否	当建立连接时使用的初始上下文。默认为 com.sun.jndi.ldap.LdapCtxFactory 。
security-realm	否	为了获得建立连接时所需的已配置的 SSLContext 而引用的安全区。

例 11.7. 添加 LDAP 转出连接

这个例子用下列属性集添加了一个转出连接：

- 搜索 DN: **cn=search,dc=acme,dc=com**
- 搜索凭证: **myPass**
- URL: **ldap://127.0.0.1:389**

第一个命令添加了安全区。

```
/host=master/core-service=management/security-  
realm=ldap_security_realm:add
```

第二个命令添加了 LDAP 连接。

```
/host=master/core-service=management/ldap-  
connection=ldap_connection/:add(search-  
credential=myPass,url=ldap://127.0.0.1:389,search-  
dn="cn=search,dc=acme,dc=com")
```

创建一个启用 LDAP 的安全区

管理接口可以针对 LDAP 服务器而不是默认的基于属性文件的安全区进行验证。LDAP 验证器将首先建立一个和远程目录服务器的连接，然后使用传入验证系统的用户名来执行搜索以找到 LDAP 记录的全限定可区分名称 (Distinguished Name , DN)。新的连接将以用户的 DN 为凭证以及用户提供的密码来建立。如果这个针对 LDAP 服务器的验证成功，DN 就被证明为有效的。

LDAP 安全区使用下列列配置属性：

connection

outbound-connections 里定义的连接的名称，用来连接 LDAP 目录。

advanced-filter

用于搜索基于提供的用户 ID 的用户的全限定过滤器。这个过滤器必须包含下列格式的变量：**{0}**。之后它会被用户提供的用户名替代。

base-dn

用户开始搜索的上下文的可区分的名称。

recursive

搜索是否应该在 LDAP 目录树里进行递归，或者只搜索指定的上下文。默认为 **false**。

user-dn

保存可区分名称的用户的属性。它会被用来测试验证。默认为 **dn**。

username-attribute

搜索用户使用的属性的名称。根据用户输入的用户名来匹配指定的属性，这个过滤器执行简单的搜索。

allow-empty-passwords

这个属性确定是否接受空的密码。这个属性的默认值是 **false**。

您必须指定 **username-filter** 或 **advanced-filter**。

advanced-filter 属性包含一个使用标准 LDAP 语法的过滤器查询，例如：

```
((&(SAMAccountName={0}))(memberOf=cn=admin,cn=users,dc=acme,dc=com))
```

例 11.8. 代表启用了 LDAP 的安全区的 XML 片段

这个例子使用了下列参数：

- **connection - ldap_connection**
- **base-dn - cn=users,dc=acme,dc=com.**
- **username-filter - attribute="sambaAccountName"**

```
<security-realm name="ldap_security_realm">
  <authentication>
    <ldap connection="ldap_connection" base-
dn="cn=users,dc=acme,dc=com">
      <username-filter attribute="sambaAccountName" />
    </ldap>
  </authentication>
</security-realm>
```



警告

确保不允许空 LDAP 密码是很重要的；除非你故意这么做，但这是严重的安全隐患。

EAP 6.1 包含一个用于 CVE-2012-5629 的补丁，它设置 LDAP 登录模块的 `allowEmptyPasswords` 选项为 `false`（如果它还没有被设置）。在旧的版本里，这个选项应该手动进行配置。

例 11.9. 添加 LDAP 安全区

下面的命令添加了一个 LDAP 验证到安全区并针对根据域里的主服务器设置其属性。

```
/host=master/core-service=management/security-  
realm=ldap_security_realm/authentication=ldap:add(base-  
dn="DC=mycompany,DC=org", recursive=true, username-  
attribute="MyAccountName", connection="ldap_connection")
```

应用新的安全区到管理接口里

在创建了安全区后，你需要在管理接口的配置里引用它。管理接口将使用安全区来进行 HTTP digest 验证。

例 11.10. 应用安全区到 HTTP 接口里

在配置完成后，你重启主机控制器，基于 web 的管理控制台将使用 LDAP 来验证用户。

```
/host=master/core-service=management/management-interface=http-  
interface/:write-attribute(name=security-  
realm,value=ldap_security_realm)
```

例 11.11. 应用安全区到 Native 接口

使用下列命令将相同的设置应用到 Native 接口：

```
/host=master/core-service=management/management-interface=native-  
interface/:write-attribute(name=security-  
realm,value=ldap_security_realm)
```

[提交 bug 报告](#)

11.8.5. 禁用 HTTP 管理接口

在受管域里，你只需要访问域控制器而不是域成员服务器上的 HTTP 接口。此外，在产品服务器上，你可能会决定禁用基于 web 的管理控制台。



注意

其他服务器，如 **JBoss Operations Network**，也使用 HTTP 接口来操作。如果你想使用这些服务，简单地禁用管理控制台自身就可以了，你可以设置 HTTP 接口的 **console-enabled** 属性为 **false**，而无需完全禁用这个接口。

```
/host=master/core-service=management/management-interface=http-interface/:write-attribute(name=console-enabled,value=false)
```

要禁用对 HTTP 接口的访问，同时也禁用对基于 Web 的管理控制台的访问，你可以将 HTTP 接口一起删除。

如果你又想再次添加这个接口，下面的 **JBoss CLI** 命令允许你读取 HTTP 接口的当前内容。

例 11.12. 读取 HTTP 接口的配置

```
/host=master/core-service=management/management-interface=http-interface/:read-resource(recursive=true,proxies=false,include-runtime=false,include-defaults=true)
{
  "outcome" => "success",
  "result" => {
    "console-enabled" => true,
    "interface" => "management",
    "port" => expression "${jboss.management.http.port:9990}",
    "secure-port" => undefined,
    "security-realm" => "ManagementRealm"
  }
}
```

要删除 HTTP 接口，请执行下列命令：

例 11.13. 删除 HTTP 接口

```
/host=master/core-service=management/management-interface=http-interface/:remove
```

要重新启用访问，执行下列命令来重新创建带有默认值的 HTTP 接口。

例 11.14. 重新创建 HTTP 接口

```
/host=master/core-service=management/management-interface=http-interface:add(console-enabled=true,interface=management,port="${jboss.management.http.port:9990}",security-realm=ManagementRealm)
```

[提交 bug 报告](#)

11.8.6. 从默认的安全区删除无提示验证

介绍

JBoss EAP 6 的默认安装包含一个用于本地管理 CLI 用户的无提示验证 (Silent Authentication) 方法。这允许本地用户无需用户名或密码验证就可以访问管理 CLI。启用这个功能是为了方便，并协助本地用户无需验证就可以运行管理 CLI 脚本。它是一个非常有用的功能，特别是对本地配置的访问通常也会让用户可以添加自己的细节或禁用安全检查。

如果需要更严格的安全检查，你也可以禁用对于本地用户的无提示验证。这可以通过删除配置文件里的 **security-realm** 部分里的 **local** 来实现。这适用于独立服务器的 **standalone.xml** 或受管域的 **host.xml**。你应该只在理解了对特定服务器配置的影响后才考虑删除 **local** 元素。

删除无提示验证的首选方法是使用管理 CLI，在下面的例子里它直接删除了 **local** 元素。

例 11.15. security-realm 里的 local 元素示例

```
<security-realms>
  <security-realm name="ManagementRealm">
    <authentication>
      <local default-user="$local"/>
      <properties path="mgmt-users.properties" relative-
to="jboss.server.config.dir"/>
    </authentication>
  </security-realm>
  <security-realm name="ApplicationRealm">
    <authentication>
      <local default-user="$local" allowed-users="*/>
      <properties path="application-users.properties" relative-
to="jboss.server.config.dir"/>
    </authentication>
    <authorization>
      <properties path="application-roles.properties" relative-
to="jboss.server.config.dir"/>
    </authorization>
  </security-realm>
</security-realms>
```

预备条件

- 第 2.1.1 节 “启动 JBoss EAP 6”
- 第 3.5.2 节 “启动管理 CLI”

过程 11.7. 从默认的安全区删除无提示验证

- 用管理 CLI 删除无提示验证
按要求从管理区和应用程序区删除 **local** 元素。
 - a. 从管理区删除 **local** 元素。
 - 对于独立服务器

```
/core-service=management/security-  
realm=ManagementRealm/authentication=local:remove
```

- 对于受管域

```
/host=HOST_NAME/core-service=management/security-  
realm=ManagementRealm/authentication=local:remove
```

b. 从应用程序区删除 **local** 元素。

- 对于独立服务器

```
/core-service=management/security-  
realm=ApplicationRealm/authentication=local:remove
```

- 对于受管域

```
/host=HOST_NAME/core-service=management/security-  
realm=ApplicationRealm/authentication=local:remove
```

结果

无提示验证从 **ManagementRealm** 和 **ApplicationRealm** 里删除了。

[提交 bug 报告](#)

11.8.7. 禁用对 JMX 子系统的远程访问

远程 JMX 连接性允许你触发 JDK 和应用程序管理操作。为了保护安装，请禁用这个功能。你可以通过删除远程连接配置或完全删除 JMX 子系统来实现这一点。JBoss CLI 命令引用了受管域配置里的 **default** 配置集。要进行修改，请修改命令的 **/profile=default** 部分。对于独立服务器，请完全删除命令的这个部分。



注意

在受管域里，远程连接器默认是从 JMX 子系统里删除的。如果是在部署时添加的，这个命令可供你参考。

例 11.16. 从 JMX 子系统删除远程连接器。

```
/profile=default/subsystem=jmx/remoting-connector=jmx/:remove
```

例 11.17. 删除 JMX 子系统

如果你使用了受管域，对你使用的每个配置集都运行这个命令。

```
/profile=default/subsystem=jmx/:remove
```

[提交 bug 报告](#)

11.8.8. 为管理接口配置安全区

管理接口使用安全区来控制验证和对 JBoss EAP 6 的配置机制的访问。本主题展示了如何阅读和配置安全区。

读取安全区的配置

这个例子展示了 **ManagementRealm** 安全区的默认配置。它使用了一个名为 **mgmt-users.properties** 的文件来保存其配置信息。

例 11.18. 默认的 ManagementRealm

```
/host=master/core-service=management/security-
realm=ManagementRealm/:read-
resource(recursive=true,proxies=false,include-runtime=false,include-
defaults=true)
{
  "outcome" => "success",
  "result" => {
    "authorization" => undefined,
    "server-identity" => undefined,
    "authentication" => {"properties" => {
      "path" => "mgmt-users.properties",
      "plain-text" => false,
      "relative-to" => "jboss.domain.config.dir"
    }}
  }
}
```

编写安全区

下面的命令创建了一个名为 **TestRealm** 的安全区并为相关的配置文件设置了目录。

例 11.19. 创建安全区

```
/host=master/core-service=management/security-realm=TestRealm/:add
/host=master/core-service=management/security-
realm=TestRealm/authentication=properties/:add(path=TestUsers.properties
, relative-to=jboss.domain.config.dir)
```

应用安全区到管理接口里

添加了安全区后，将其名称作为引用提供给管理接口。

例 11.20. 在管理接口里添加一个安全区

```
/host=master/core-service=management/management-interface=http-
interface/:write-attribute(name=security-realm,value=TestRealm)
```

[提交 bug 报告](#)

11.9. 用基于角色的访问控制来保护管理接口

11.9.1. 关于基于角色的访问控制（Role-Based Access Control，RBAC）

关于基于角色的访问控制（RBAC）是一个指定管理用户的一系列权限的机制。它允许多个用户共享管理 JBoss EAP 6.3 服务器的职责而不是拥有不受限制的权限。通过对管理用户的“职责分离”，JBoss EAP 6.3 使机构可以轻易地在个人和组之间分配职责，从而避免分配不必要的权限。这既确保了服务器和数据最大可能的安全性，也提供了配置、部署和管理的灵活性。

JBoss EAP 6.3 里的关于基于角色的访问控制是通过角色权限和约束的组合来实现的。

它提供了 7 个预定义的角色，每个都有不同的固定权限。这些预定义的角色是：Monitor、Operator、Maintainer、Deployer、Auditor、Administrator 和 SuperUser。每个管理用户都分配了一个或多个角色，这些角色指定了管理服务器时用户被允许做的事情。

[提交 bug 报告](#)

11.9.2. 管理控制台和 CLI 里基于角色的访问控制

当启用了基于角色的访问控制（Role-Based Access Control，RBAC）时，用户可以访问的资源以及用资源属性进行的操作是由所分配的角色来决定的。

管理控制台

在管理控制台里，根据分配给用户的角色的权限，有些控件和视图是禁用的（灰色）或不可见的。

如果你没有对某个资源属性的读权限，该权限将在控制台里显示为空白。例如，多数角色无法读取数据源的用户名和密码字段。

如果你没有对某个资源属性的写权限，该权限将在资源编辑表单里显示为禁用（灰色）。如果你没有对这个资源的写权限，那么整个编辑表单都不会出现。

如果用户没有对某个资源或属性的访问权限（就是对于这个角色来说是“不可寻址的”），那它们不会出现在该用户的控制台里。其中一个例子是，访问控制系统自身在默认情况下只对一些角色可见。

管理 CLI 或 API

启用了 RBAC 后，使用 `jboss-cli.sh` 工具或管理 API 的用户在 API 里会遇到稍许不同的行为。

无法读取的资源 and 属性将从结果里过滤。如果被过滤的内容是角色可以寻址的，那么它们的名称将列为结果里 `response-headers` 部分的 `filtered-attributes`。如果它们是无法被寻址的，那就不会被列出。

试图访问不能寻址的资源将导致 `resource not found` 错误。

如果用户试图写入或读取他们可以寻址的资源但缺乏对应的读写权限，就会返回 `Permission Denied` 错误。

[提交 bug 报告](#)

11.9.3. 支持的验证模式

基于角色的访问控制（RBAC）可以和 JBoss EAP 6.3 附带的标准验证提供者一起使用。它们是：`username/password`、`client certificate` 和 `local user`。

Username/Password

用户通过用户名和密码组合根据 `mgmt-users.properties` 文件或 LDAP 服务器来进行检验。

Client Certificate

使用信任库。

Local User

如果服务器运行在相同的服务器上，`jboss-cli.sh` 将自动验证为本地用户。在默认情况下，本地用户是 **SuperUser** 组的成员。

不管使用哪个提供者，JBoss EAP 负责将角色分配给用户。然而，当用 `mgmt-users.properties` 文件或 LDAP 服务器来验证时，这些系统可以提供用户组信息。JBoss EAP 也可以使用这些信息来为用户分配角色。

[提交 bug 报告](#)

11.9.4. 标准角色

JBoss EAP 6 提供 7 个预定义的用户角色：**Monitor**、**Operator**、**Maintainer**、**Deployer**、**Auditor**、**Administrator** 和 **SuperUser**。每个角色都有不同的一套权限以用于专有的用例。**Monitor**、**Operator**、**Maintainer**、**Administrator** 和 **SuperUser** 都是在前者之上构建的，其权限一次递增。**Auditor** 和 **Deployer** 角色分别与 **Monitor** 和 **Maintainer** 角色类似，但都具有一些其他的特殊权限和限制。

Monitor

充当 **Monitor** 角色的用户具有最小的权限且只能读取服务器的当前配置和状态。这个角色是为了那些需要跟踪和报告服务器性能的用户而设计的。

Monitor 角色既不能修改服务器配置也不能访问敏感数据或操作。

Operator

Operator 角色扩展了 **Monitor** 角色，它添加了修改服务器的运行时状态的能力。这表示 **Operator** 可以重载并关闭服务器，也可以暂停和恢复 JMS 目的地。**Operator** 角色对于负责应用服务器的物理或虚拟主机的用户是很理想的，他们可以确保有需要时关闭和重启服务器。

Operator 角色既不能修改服务器配置也不能访问敏感数据或操作。

Maintainer

Maintainer 角色可以查看和修改运行时状态以及除了敏感数据和操作之外的所有配置。**Maintainer** 角色是普通用途的角色，它不能访问敏感数据和操作。**Maintainer** 角色赋予用户几乎完整的管理服务器的权限，但不能访问密码和其他敏感信息。

Maintainer 角色不能访问敏感数据或操作。

Administrator

Administrator 角色具有对服务器上除了审计日志系统以外的所有资源和操作的无限制的访问权限。**Administrator** 是唯一（除了 **SuperUser**）可以访问敏感数据和操作的角色。这个角色也可以配置访问控制系统。只有处理敏感数据或配置用户和角色时才要求 **Administrator** 角色。

Administrator 角色不能访问审计日志系统且不能将自己修改为 Auditor 或 SuperUser 角色。

SuperUser

SuperUser 角色没有任何限制，它可以访问任何服务器的资源和操作，包括审计日志系统。这个角色等同于以前的 JBoss EAP 6 版本（6.0 和 6.1）里的管理员用户。如果禁用了 RBAC，所有的管理用户都会拥有和 SuperUser 角色相同的权限。

Deployer

Deployer 角色具有和 Monitor 相同的角色，但它可以修改部署的配置和状态以及其他启用为应用程序资源的资源类型。

Auditor

Auditor 角色具有 Monitor 角色的所有权限，它也可以查看（但不能修改）敏感数据，且具有对审计日志系统的完全权限。Auditor 是除了 SuperUser 之外唯一能够访问审计日志系统的角色。

Auditor 角色不能修改敏感数据或资源。它只有读的权限。

[提交 bug 报告](#)

11.9.5. 关于角色权限

角色具有的权限定义了它能做的事情。并非每个角色都具有每个权限。值得注意的是，SuperUser 具有所有权限，而 Monitor 的权限最小。

每个权限都可以为某个类别的资源赋予读和写权限。

这些类别是：运行时状态、服务器配置、敏感数据、审计日志和访问控制系统。

表 11.5 “角色权限矩阵” 总结了每个角色的权限。

表 11.5. 角色权限矩阵

	Monitor	Operator	Maintainer	Deployer	Auditor	Administrator	SuperUser
读取配置和状态	X	X	X	X	X	X	X
读取敏感数据 [2]					X	X	X
修改敏感数据 [2]						X	X
读/修改审计日志					X		X
修改运行时状态		X	X	X[1]		X	X

修改持久性配置			X	X[1]		X	X
读/修改访问控制						X	X

[1] 权限限于应用程序资源。

[2] 哪些资源被当作 "敏感数据" 是使用敏感性约束来配置的。

[提交 bug 报告](#)

11.9.6. 关于约束

约束 (Constraint) 是对于指定资源列表的访问控制配置的集合。RBAC 系统使用了约束和角色权限的组合来确定某个用户是否可以执行管理动作。

约束分成三个类别：应用程序 (application) 和敏感性 (sensitivity) 和 Vault 表达式。

应用程序约束

应用程序约束定义了可以被具有部署角色的用户访问的资源和属性集合。在默认情况下，唯一被启用的应用程序约束是 **core**，它包含部署、部署重叠。应用程序约束也包含在 **datasources**、**logging**、**mail**、**messaging**、**naming**、**resource-adapters** 和 **security** 里 (默认不会启用)。这些约束允许 **Deployer** 用户不仅可以部署应用程序，还可以配置和维护应用程序要求的资源。

应用程序约束的配置位于管理 API 里的 **/core-service=management/access=authorization/constraint=application-classification**。

敏感性约束

敏感性约束定义了被认为是“敏感的”的资源的集合。敏感的资源通常是某种机密的内容，如密码，或者对服务器的操作有着重大影响的东西，如网络、**JVM** 配置或系统属性。访问控制系统本身也被视作敏感的资源。

唯一对敏感资源具有写权限的角色是 **Administrator** 和 **SuperUser**。**Auditor** 角色只能读取敏感资源。其他角色都没有访问权限。

应用程序约束的配置位于管理 API 里的 **/core-service=management/access=authorization/constraint=sensitivity-classification**。

库表达式约束

库表达式 (Vault Expression) 约束定义了读/写库表达式是否被视作敏感操作。在默认情况下，这两者都是敏感操作。

库表达式约束的配置位于管理 API 的 **/core-service=management/access=authorization/constraint=vault-expression**。

目前你不能在管理控制台配置约束。

[提交 bug 报告](#)

11.9.7. 关于 JMX 和基于角色的访问控制

基于角色的访问控制以三种方式应用于 JMX：

1. JBoss EAP 6 的 Management API 开放为 JMX Management Bean。这些 Management Bean 被称为 "core mbeans"，对它们的访问和过滤同底层的 Management API 是一模一样的。
2. JMX 子系统的写权限被配置为“敏感的”。这表示只有 Administrator 和 SuperUser 角色可以修改这个子系统。而 Auditor 角色可以读取子系统的配置。
3. 在默认情况下，部署的应用程序和服务（non-core mbeans）注册的 Management Bean 可以被管理用户访问，但只有 Maintainer、Operator、Administrator、SuperUser 角色可以写入它。

[提交 bug 报告](#)

11.9.8. 配置基于角色的访问控制

11.9.8.1. RBAC 配置任务概述

启用了 RBAC 后，只有具有 Administration 或 SuperUser 角色的用户可以查看和修改访问控制系统。

管理控制台提供了下列常见 RBAC 任务的界面：

- 查看和配置用户分配（或排斥）了哪些角色
- 查看和配置组分配（或排斥）了哪些角色
- 查看每个角色的组和用户成员资格。
- 配置每个角色默认成员资格。
- 常见具有作用域的角色

管理 CLI 提供了对整个访问控制系统的访问。这表示在管理控制台里可以完成的任何事情都可以在这里完成，而且你可以用 CLI 执行一些访问控制台无法完成的任务。

在 CLI 里也可以执行下列任务：

- 启用和禁用 RBAC
- 修改权限组合策略
- 配置应用程序资源和资源敏感性约束

[提交 bug 报告](#)

11.9.8.2. 启用基于角色的访问控制

在默认情况下，RBAC 是被禁用的。将提供者属性从 **simple** 修改为 **rbac** 就可以启用它。你可以用 **jboss-cli.sh** 工具来完成，或者当服务器下线时编辑服务器配置 XML 文件。如果在运行的服务器上禁用或启用 RBAC，在生效前必须重载服务器配置。

启用后，它只能由具有 Administrator 或 SuperUser 角色的用户禁用。在默认情况下，如果 **jboss-cli.sh** 运行在和服务器相同的主机上，它是以 **SuperUser** 角色运行的。

过程 11.8. 启用 RBAC

- 要用 **jboss-cli.sh** 启用 RBAC，请使用访问授权资源的 **write-attribute** 操作来设置提供者的属性为 **rbac**。

```
/core-service=management/access=authorization:write-attribute(name=provider, value=rbac)
```

```
[standalone@localhost:9999 /] /core-
service=management/access=authorization:write-
attribute(name=provider, value=rbac)
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
[standalone@localhost:9999 /] /:reload
{
  "outcome" => "success",
  "result" => undefined
}
```

过程 11.9. 禁用 RBAC

- 要用 **jboss-cli.sh** 禁用 RBAC，请使用访问授权资源的 **write-attribute** 操作来设置提供者的属性为 **simple**。

```
/core-service=management/access=authorization:write-attribute(name=provider, value=simple)
```

```
[standalone@localhost:9999 /] /core-
service=management/access=authorization:write-
attribute(name=provider, value=simple)
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
[standalone@localhost:9999 /] /:reload
{
  "outcome" => "success",
  "result" => undefined
}
```

如果服务器已下线，你可以编辑 XML 配置文件来启用或禁用 RBAC。你可以找到 **management** 元素下的 **access-control** 元素里的 **provider** 属性，设置为 **rbac** 则启用，**simple** 则禁用 RBAC。

```
<management>
```



```

        <access-control provider="rbac">
            <role-mapping>
                <role name="SuperUser">
                    <include>
                        <user name="$local"/>
                    </include>
                </role>
            </role-mapping>
        </access-control>

    </management>

```

[提交 bug 报告](#)

11.9.8.3. 修改权限组合策略

权限组合策略（Permission Combination Policy）定义了用户被分配了多个角色时如何确定权限。它可以设置为 **permissive** 或 **rejecting**。默认选项是 **permissive**。

当设置为 **permissive** 时，如果任何角色被分配给用户，那这个角色许可的动作将会被允许执行。

当策略被设置为 **rejecting** 时，如果用户分配了多个角色，将不允许任何动作。这意味着当策略为 **rejecting** 时，每个用户应该只分配一个角色。当策略为 **rejecting** 时，具有多个角色的用户将不能使用管理控制台或 **jboss-cli.sh** 工具。

权限组合策略是通过设置 **permission-combination-policy** 属性为 **permissive** 或 **rejecting** 来配置的。这可以用 **jboss-cli.sh** 工具或编辑服务器配置 XML 文件（如果服务器已下线）来完成。

过程 11.10. 设置权限组合策略

- 使用访问授权资源的 **write-attribute** 操作来设置 **permission-combination-policy** 为所需的策略名称。

```

/core-service=management/access=authorization:write-
attribute(name=permission-combination-policy, value=POLICYNAME)

```

有效的策略名称是 **rejecting** 和 **permissive**。

```

[standalone@localhost:9999 /] /core-
service=management/access=authorization:write-
attribute(name=permission-combination-policy, value=rejecting)
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]

```

如果服务器已下线，你可以编辑 XML 配置文件来修改权限组合策略。为此，你需要编辑 **access-control** 元素的 **permission-combination-policy** 属性。

```

<access-control provider="rbac" permission-combination-policy="rejecting">
    <role-mapping>
        <role name="SuperUser">
            <include>
                <user name="$local"/>
            </include>
        </role>
    </role-mapping>
</access-control>

```



```

    </role>
  </role-mapping>
</access-control>

```

[提交 bug 报告](#)

11.9.9. 管理角色

11.9.9.1. 关于角色成员资格

当启用了基于角色的控制（RBAC）时，管理用户被允许做的事情是由所分配的角色来决定的。JBoss EAP 6.3 使用了基于用户和组成员资格的包含（**include**）和排除（**exclude**）列表来确定用户具有哪些角色。

用户会被认为分配了角色，如果：

1. 用户被：
 - 列在角色包含列表里，或者
 - 是列在角色包含列表里的组的成员。
2. 用户没有被：
 - 列在角色排除列表里，或者
 - 是列在角色排除列表里的组的成员。

排除列表优先级高于包含列表。

用户和组的角色包含和排除也可以使用管理控制台和管理 CLI 工具来完成。

只有具有 **SuperUser** 或 **Administrator** 角色的用户才能执行这个配置。

[提交 bug 报告](#)

11.9.9.2. 配置用户和角色的分配

包含或排除用户的角色可以通过管理控制台和 **jboss-cli.sh** 进行配置。本节只展示如何使用管理控制台来完成。

只有具有 **SuperUser** 或 **Administrator** 角色的用户才能执行这些配置。

在管理控制台里可通过下列步骤来配置用户和角色：

1. 登陆到管理控制台。
2. 点击 **Administration** 标签页。
3. 展开 **Access Control** 菜单并选择 **Role Assignment**。
4. 选择 **USERS** 标签页。

过程 11.11. 为用户创建新的角色分配

1. 登陆到管理控制台。

2. 进入 **Role Assignment** 部分的 **Users** 标签页。
3. 点击用户列表右上角的 **Add** 按钮。**Add User** 对话框会出现。

Add User

User:

Realm:

Type: Include

Roles:

	Name
<input type="checkbox"/>	Administrator
<input type="checkbox"/>	Auditor
<input type="checkbox"/>	Deployer
<input type="checkbox"/>	Maintainer
<input type="checkbox"/>	Monitor
<input type="checkbox"/>	Operator
<input type="checkbox"/>	SuperUser

<< < 1-7 of 8 > >>

Cancel Save

图 11.1. 『Add User』对话框

4. 指定用户名，可选择输入区名。
5. 选择是包含（include）还是排除（exclude）。
6. 点击要包含或排除的角色旁的复选框。您可以使用 Ctl 键（OSX 上的 Command 键）来选择多个选项。

7. 点击 **Save** 完成。

保存成功后，**Add User** 对话框将会关闭，用户列表将会更新以反映所作的修改。如果不成功则会显示 **Failed to save role assignment** 消息。

过程 11.12. 更新用户的角色分配

- 1. 登陆到管理控制台。
- 2. 进入 **Role Assignment** 部分的 **Users** 标签页。
- 3. 从列表里选择用户。
- 4. 点击 **Edit**。『**Selection**』面板将进入编辑模式。

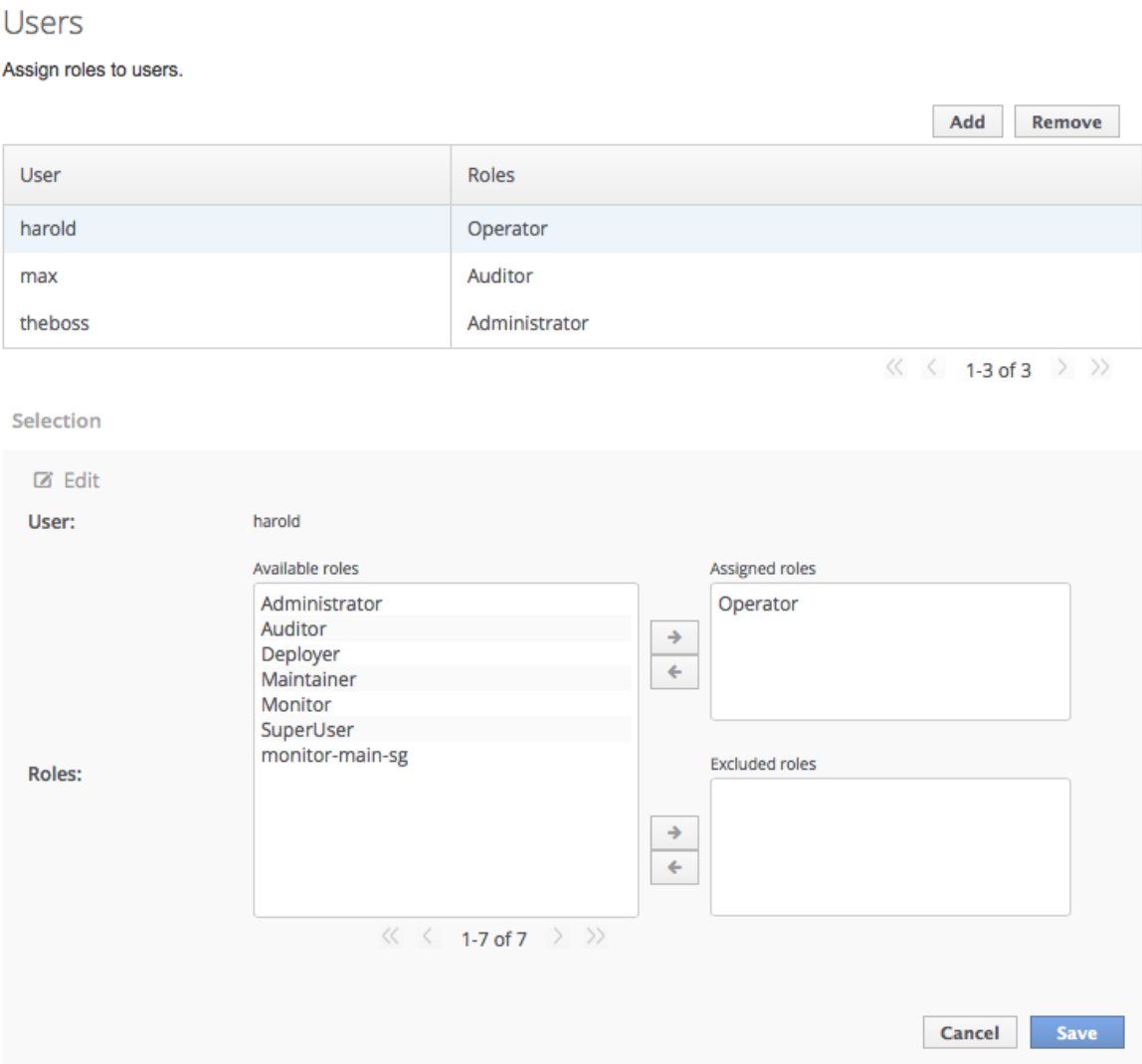


图 11.2. 『Selection』面板的编辑视图

在这里你可以添加和删除所分配或排除的角色。

- 1. 要添加被分配的角色，从左侧的『**Available roles**』列表里进行选择并点击『**Assigned roles**』列表旁的右箭头。角色将从可用列表移至『**Assigned roles**』列表。

2. 要删除被分配的角色，从右侧的『Assigned roles』列表里选择要删除的角色，并点击旁边的左箭头。角色将从『Assigned roles』列表移至『Available roles』列表。
3. 要添加被排除的角色，从左侧的『Available roles』列表里进行选择并点击『Excluded roles』列表旁的右箭头。角色将从可用列表移至『Excluded roles』列表。
4. 要删除被排除的角色，从右侧的『Excluded roles』列表里选择要删除的角色，并点击旁边的左箭头。角色将从『Excluded roles』列表移至『Available roles』列表。
5. 点击 **Save** 完成。

成功后，编辑视图将会关闭，用户列表将会更新以反映所作的修改。如果不成功则会显示 **Failed to save role assignment** 消息。

过程 11.13. 删除用户的角色分配

1. 登录到管理控制台。
2. 进入『Role Assignment』部分的 **Users** 标签页。
3. 从列表里选择用户。
4. 点击 **Remove**。Remove Role Assignment 配置提示将会出现。
5. 点击 **Confirm**。

成功后，用户将不会再出现在用户角色分配的列表里。



重要

从角色分配列表里删除用户并不会从系统删除这个用户，也不能保证没有角色被分配给这个用户。角色仍可能通过组成员资格分配给用户。

[提交 bug 报告](#)

11.9.9.3. 用 `jboss-cli.sh` 配置用户角色分配

包含或排除用户的角色可以通过管理控制台和 `jboss-cli.sh` 进行配置。本节只展示如何使用 `jboss-cli.sh` 来完成。

映射用户/组到角色的配置位于 management API 里的 `/core-service=management/access=authorization as role-mapping` 元素。

只有具有 SuperUser 或 Administrator 角色的用户才能执行这个配置。

过程 11.14. 查看角色分配配置

1. 使用 `:read-children-names` 操作来获取配置角色的完整列表：

```
/core-service=management/access=authorization:read-children-
names(child-type=role-mapping)
```

```
[standalone@localhost:9999 access=authorization] :read-children-
names(child-type=role-mapping)
```

```
{
  "outcome" => "success",
  "result" => [
    "ADMINISTRATOR",
    "DEPLOYER",
    "MAINTAINER",
    "MONITOR",
    "OPERATOR",
    "SuperUser"
  ]
}
```

2. 使用指定 **role-mapping** 的 **read-resource** 操作来获取某个角色的完整细节：

```
/core-service=management/access=authorization/role-
mapping=ROLENAME:read-resource(recursive=true)
```

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=ADMINISTRATOR:read-resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "include-all" => false,
    "exclude" => undefined,
    "include" => {
      "user-theboss" => {
        "name" => "theboss",
        "realm" => undefined,
        "type" => "USER"
      },
      "user-harold" => {
        "name" => "harold",
        "realm" => undefined,
        "type" => "USER"
      },
      "group-SysOps" => {
        "name" => "SysOps",
        "realm" => undefined,
        "type" => "GROUP"
      }
    }
  }
}
[standalone@localhost:9999 access=authorization]
```

过程 11.15. 添加新的角色

这个过程展示了如何添加角色的 **role-mapping** 条目。这必须在角色可以被配置前完成。

- 使用 **add** 操作来添加新的角色配置。

```
/core-service=management/access=authorization/role-
mapping=ROLENAME:add
```

ROLENAME 是新映射使用的角色的名称。

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR:add
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

过程 11.16. 添加包含在角色里的用户

这个过程展示了如何添加用户到角色的包含列表里。

如果角色的配置还未完成，那你必须先设置 **role-mapping** 条目。

- 请使用 **add** 操作来添加用户到角色的包含列表里。

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/include=ALIAS:add(name=USERNAME, type=USER)
```

ROLENAME 是被配置的角色名称。

ALIAS 是这个映射的唯一名称。红帽推荐你对别名使用命名规则，如 **user-USERNAME**。

USERNAME 是添加到包含列表里的用户的名称。

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/include=user-max:add(name=max, type=USER)
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

过程 11.17. 添加角色所排除的用户

这个过程展示了如何添加用户到角色的排除列表里。

如果角色的配置还未完成，那你必须先设置 **role-mapping** 条目。

- 请使用 **add** 操作来添加用户到角色的排除列表里。

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude=ALIAS:add(name=USERNAME, type=USER)
```

ROLENAME 是被配置的角色名称。

USERNAME 是添加到排除列表里的用户的名称。

ALIAS 是这个映射的唯一名称。红帽推荐你对别名使用命名规则，如 **user-USERNAME**。

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/exclude=user-max:add(name=max, type=USER)
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

过程 11.18. 删除用户的角色包含配置

这个过程展示了如何从角色映射里删除用户包含条目。

- 请使用 **remove** 操作来删除这个条目。

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/include=ALIAS:remove
```

ROLENAME 是被配置的角色名称。

ALIAS 是这个映射的唯一名称。红帽推荐你对别名使用命名规则，如 **user-USERNAME**。

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/include=user-max:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

从包含列表里删除用户并不会从系统删除这个用户，也不能保证角色不会被分配给这个用户。这个角色仍可能根据组成员资格分配给它。

过程 11.19. 删除用户的角色排除配置

这个过程展示了如何从角色映射里删除用户排除条目。

- 请使用 **remove** 操作来删除这个条目。

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude=ALIAS:remove
```

ROLENAME 是被配置的角色名称。

ALIAS 是这个映射的唯一名称。红帽推荐你对别名使用命名规则，如 **user-USERNAME**。

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/exclude=user-max:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

从排除列表里删除用户并不会从系统删除这个用户，也不能保证角色被分配给这个用户。这个角色仍可能根据组成员资格被排除。

[提交 bug 报告](#)

11.9.9.4. 关于角色和用户组

使用 **mgmt-users.properties** 文件或 LDAP 服务器验证的用户可以是用户组的成员。用户组是可以分配给一个或多个用户的任意标签。

你可以配置 RBAC 系统根据用户所在的用户组自动分配角色给用户。它也可以根据用户组成员资格来排除用户。

当使用 **mgmt-users.properties** 文件时，组信息保存在 **mgmt-groups.properties** 文件里。当使用 LDAP 时，组信息保存在 LDAP 服务器里并由负责 LDAP 服务器的人员来维护。

[提交 bug 报告](#)

11.9.9.5. 配置组角色的分配

角色可以根据用户的用户组成员资格分配给用户。

角色包含或排除组可以通过管理控制台和 **jboss-cli.sh** 进行配置。本节只展示如何使用管理控制台来完成。

只有具有 **SuperUser** 或 **Administrator** 角色的用户才能执行这些配置。

在管理控制台里可通过下列步骤来配置组和角色：

1. 登陆到管理控制台。
2. 点击 **Administration** 标签页。
3. 展开 **Access Control** 菜单并选择 **Role Assignment**。
4. 选择 **GROUPS** 标签页。

过程 11.20. 为组创建新的角色分配

1. 登陆到管理控制台
2. 进入 **Role Assignment** 部分的 **GROUPS** 标签页。
3. 点击用户列表右上角的 **Add** 按钮。**Add Group** 对话框会出现。

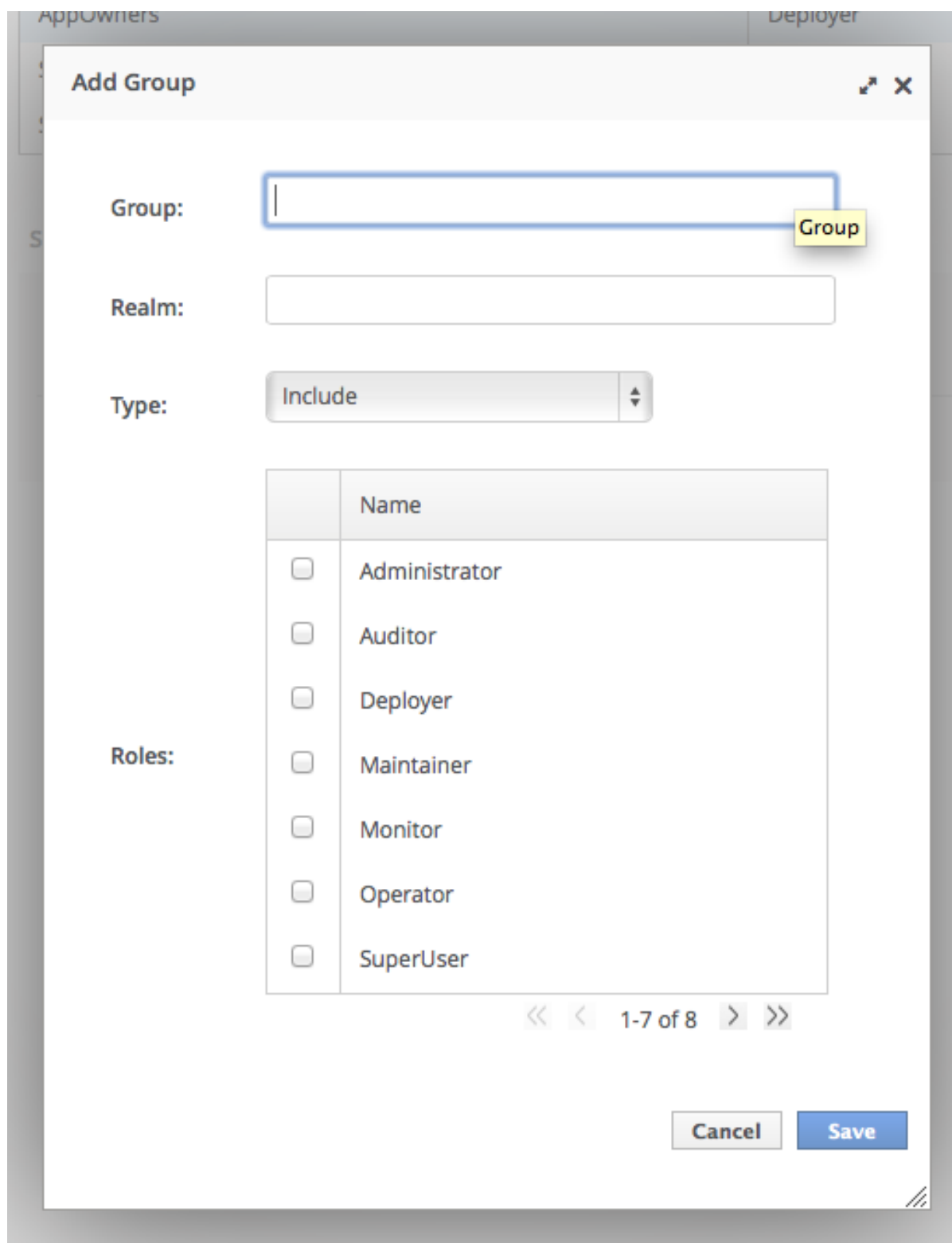


图 11.3. 『Add Group』对话框

4. 指定组名，可选择输入区名。
5. 选择是包含（include）还是排除（exclude）。
6. 点击要包含或排除的角色旁的复选框。您可以使用 Ctl 键（OSX 上的 Command 键）来选择多个选项。
7. 点击**Save**完成。

保存成功后，**Add Group** 对话框将会关闭，组列表将会更新以反映所作的修改。如果不成功则会显示 **Failed to save role assignment** 消息。

过程 11.21. 更新组的角色分配

- 1. 登录到管理控制台。
- 2. 进入『Role Assignment』的 **GROUPS** 标签页。
- 3. 从列表里选择组。
- 4. 点击『Edit』。『Selection』面板将进入编辑模式。

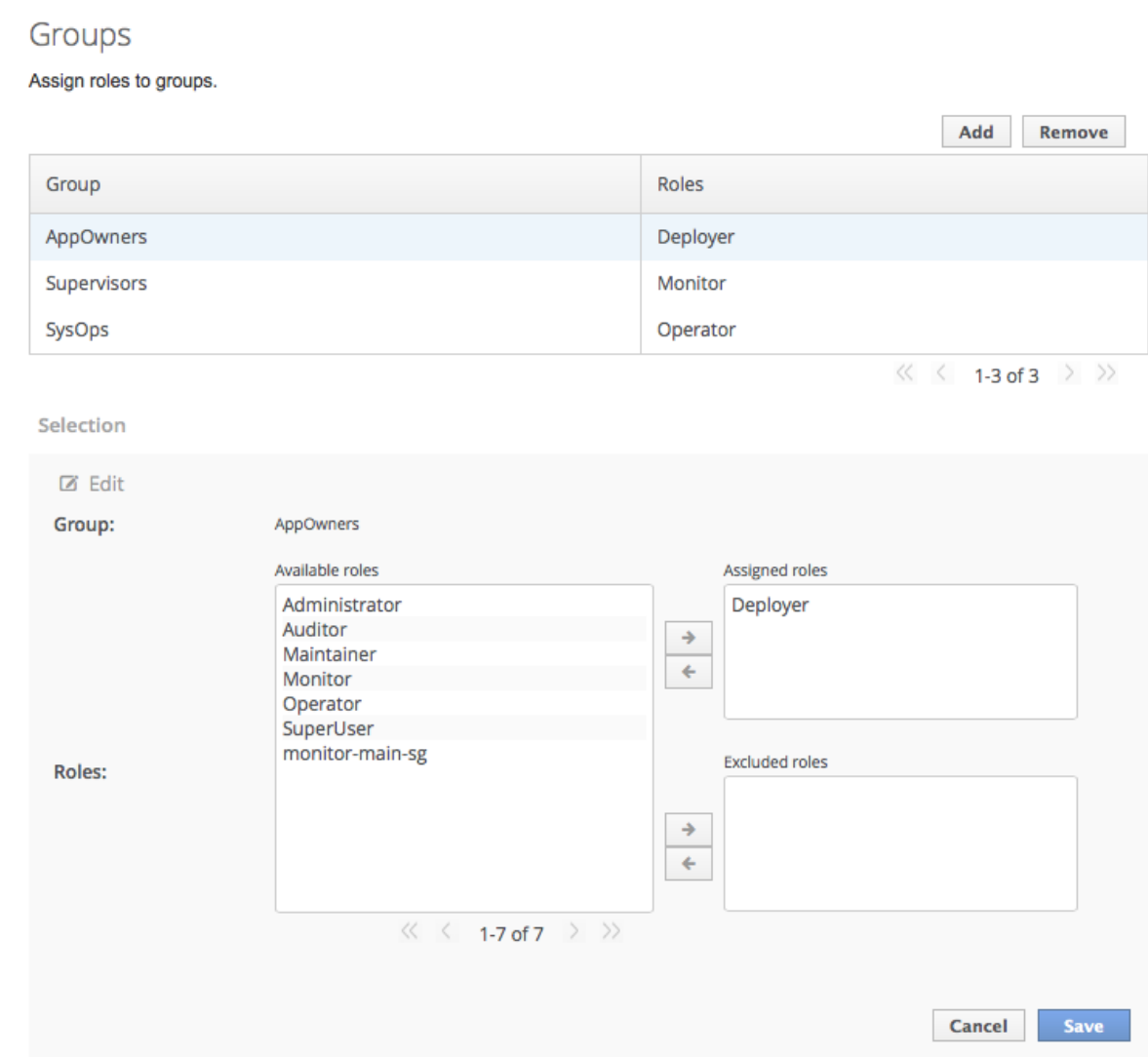


图 11.4. 『Selection』视图编辑模式

在这里你可以添加和删除组里分配或排除的角色：

- 要添加被分配的角色，从左侧的『Available roles』列表里进行选择并点击『Assigned roles』列表旁的右箭头。角色将从可用列表移至『Assigned roles』列表。
- 要删除被分配的角色，从右侧的『Assigned roles』列表里选择要删除的角色，并点击旁边的左箭头。角色将从『Assigned roles』列表移至『Available roles』列表。

- 要添加被排除的角色，从左侧的『Available roles』列表里进行选择并点击『Excluded roles』列表旁的右箭头。角色将从可用列表移至『Excluded roles』列表。
- 要删除被排除的角色，从右侧的『Excluded roles』列表里选择要删除的角色，并点击旁边的左箭头。角色将从『Excluded roles』列表移至『Available roles』列表。

5. 点击 **Save** 完成。

成功后，编辑视图将会关闭，组列表将会更新以反映所作的修改。如果不成功则会显示 **Failed to save role assignment** 消息。

过程 11.22. 删除组的角色分配

1. 登陆到管理控制台。
2. 进入 **Role Assignment** 的 **GROUPS** 标签页。
3. 从列表里选择组。
4. 点击 **Remove**。 **Remove Role Assignment** 配置提示将会出现。
5. 点击 **Confirm** 确认。

成功后，组将不会再出现在用户角色分配的列表里。

从角色分配列表里删除组并不会从系统删除这个用户组，也不能保证没有角色被分配给这个组。每个组成员仍可能被直接分配角色。

[提交 bug 报告](#)

11.9.9.6. 用 `jboss-cli.sh` 配置组角色

角色包含或排除组可以通过管理控制台和 `jboss-cli.sh` 进行配置。本节只展示如何使用 `jboss-cli.sh` 工具来完成。

映射用户/组到角色的配置位于 management API 里的 `/core-service=management/access=authorization as role-mapping` 元素。

只有具有 **SuperUser** 或 **Administrator** 角色的用户才能执行这些配置。

过程 11.23. 查看组角色分配配置

1. 使用 `read-children-names` 操作来获取配置角色的完整列表：

```
/core-service=management/access=authorization:read-children-
names(child-type=role-mapping)
```

```
[standalone@localhost:9999 access=authorization] :read-children-
names(child-type=role-mapping)
{
  "outcome" => "success",
  "result" => [
    "ADMINISTRATOR",
    "DEPLOYER",
    "MAINTAINER",
```

```

        "MONITOR",
        "OPERATOR",
        "SuperUser"
    ]
}

```

2. 使用指定 **role-mapping** 的 **read-resource** 操作来获取某个角色的完整细节：

```

/core-service=management/access=authorization/role-
mapping=ROLENAME:read-resource(recursive=true)

[standalone@localhost:9999 access=authorization] ./role-
mapping=ADMINISTRATOR:read-resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
        "include-all" => false,
        "exclude" => undefined,
        "include" => {
            "user-theboss" => {
                "name" => "theboss",
                "realm" => undefined,
                "type" => "USER"
            },
            "user-harold" => {
                "name" => "harold",
                "realm" => undefined,
                "type" => "USER"
            },
            "group-SysOps" => {
                "name" => "SysOps",
                "realm" => undefined,
                "type" => "GROUP"
            }
        }
    }
}
[standalone@localhost:9999 access=authorization]

```

过程 11.24. 添加新的角色

这个过程展示了如何添加角色的 **role-mapping** 条目。这必须在角色可以被配置前完成。

- 使用 **add** 操作来添加新的角色配置。

```

/core-service=management/access=authorization/role-
mapping=ROLENAME:add

[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR:add
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]

```

过程 11.25. 添加包含在角色里的组

这个过程展示了如何添加组到角色的包含列表里。

如果角色的配置还未完成，那你必须先设置 **role-mapping** 条目。

- 请使用 **add** 操作来添加组到角色的包含列表里。

```
/core-service=management/access=authorization/role-  
mapping=ROLENAME/include=ALIAS:add(name=GROUPNAME, type=GROUP)
```

ROLENAME 是被配置的角色名称。

GROUPNAME 是添加到包含列表里的组的名称。

ALIAS 是这个映射的唯一名称。红帽推荐你对别名使用命名规则，如 **group-*GROUPNAME***。

```
[standalone@localhost:9999 access=authorization] ./role-  
mapping=AUDITOR/include=group-investigators:add(name=investigators,  
type=GROUP)  
{"outcome" => "success"}  
[standalone@localhost:9999 access=authorization]
```

过程 11.26. 添加角色所排除的组

这个过程展示了如何添加组到角色的排除列表里。

如果角色的配置还未完成，那你必须先创建 **role-mapping** 条目。

- 请使用 **add** 操作来添加组到角色的排除列表里。

```
/core-service=management/access=authorization/role-  
mapping=ROLENAME/exclude=ALIAS:add(name=GROUPNAME, type=GROUP)
```

ROLENAME 是被配置的角色名称。

GROUPNAME 是添加到包含列表里的组的名称。

ALIAS 是这个映射的唯一名称。红帽推荐你对别名使用命名规则，如 **group-*GROUPNAME***。

```
[standalone@localhost:9999 access=authorization] ./role-  
mapping=AUDITOR/exclude=group-supervisors:add(name=supervisors,  
type=USER)  
{"outcome" => "success"}  
[standalone@localhost:9999 access=authorization]
```

过程 11.27. 删除组的角色包含配置

这个过程展示了如何从角色映射里删除组包含条目。

- 请使用 **remove** 操作来删除这个条目。

```
/core-service=management/access=authorization/role-  
mapping=ROLENAME/include=ALIAS:remove
```

ROLENAME 是被配置的角色名称。

ALIAS 是这个映射的唯一名称。红帽推荐你对别名使用命名规则，如 **group-*GROUPNAME***。

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/include=group-investigators:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

从包含列表里删除组并不会从系统删除这个组，也不能保证角色不会被分配给这个组。这个角色仍可能分配给组里的用户。

过程 11.28. 删除组的角色排除配置

这个过程展示了如何从角色映射里删除组排除条目。

- 请使用 **remove** 操作来删除这个条目。

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude=ALIAS:remove
```

ROLENAME 是被配置的角色名称。

ALIAS 是这个映射的唯一名称。红帽推荐你对别名使用命名规则，如 **group-*GROUPNAME***。

```
[standalone@localhost:9999 access=authorization] ./role-
mapping=AUDITOR/exclude=group-supervisors:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

从排除列表里删除组并不会从系统删除这个组，也不能保证角色被分配给这个组。这个角色仍可能根据组成员资格被排除。

[提交 bug 报告](#)

11.9.9.7. 关于用 LDAP 进行授权和组加载

在 LDAP 目录里，有些条目用于用户帐号而有些条目用于组，并通过使用属性来进行交叉引用。有些属性是从用户帐号引用组条目，或者是组上的属性引用作为组成员的用户。在一些服务器上，这两种交叉引用的形式都存在。

用户使用简单的用户名通过服务器来验证也是很常见的，搜索组成员信息取决于使用的目录服务器。你可以使用简单名称来执行搜索，也可以用目录里的用户条目的标识名来执行。

用户连接服务器的验证步骤总是先发生的，一旦它已经决定用户成功验证，服务器将开始加载用户组。因为验证和授权步骤都使用到 LDAP 服务器的连接，安全区包含一个优化办法，也就是任何用于验证的连接都将被组加载步骤重用。如下面的配置步骤所展示的，你可以在授权部分定义角色来将用户的简单名称转换为标识名，这可能复制在验证步骤发生的搜索，所以，如果用户名到标识名的搜索已被执行，那么这个搜索的结果就可以被缓存和重用而无需再次进行搜索。

```
<authorization>
  <ldap connection="...">
    <username-to-dn> <!-- OPTIONAL -->
      <!-- Only one of the following. -->
```

```

        <username-is-dn />
        <username-filter base-dn="..." recursive="..." user-dn-
attribute="..." attribute="..." />
        <advanced-filter base-dn="..." recursive="..." user-dn-
attribute="..." filter="..." />
    </username-to-dn>
    <group-search group-name="..." iterative="..." group-dn-
attribute="..." group-name-attribute="..." >
        <!-- One of the following -->
        <group-to-principal base-dn="..." recursive="..." search-
by="...">
            <membership-filter principal-attribute="..." />
        </group-to-principal>
        <principal-to-group group-attribute="..." />
    </group-search>
</ldap>
</authorization>

```



重要

有些例子中的属性是用默认值设置的，在此出现的目的是为了说明而已。当服务器进行持久化时，这些包含默认值的属性将从配置里删除。

username-to-dn

如之前所述，有时候你需要在授权配置里定义如何从被验证的用户的用户名映射到 LDAP 目录里条目的标识名。**username-to-dn** 元素是关于它的定义的，只有下列两者都为 **true** 才要求定义这个元素：

- 验证步骤不是通过 LDAP 进行的。
- 组搜索在搜索过程中使用标识名。

1:1 username-to-dn

这个是最基本的配置形式，用于指定远程用户输入的用户名实际上是标识名。

```

<username-to-dn>
    <username-is-dn />
</username-to-dn>

```

因为这是定义的 1:1 的映射，所以没有其他可能的配置。

username-filter

下一个选项和上面验证步骤描述的选项非常类似。指定的属性按照提供的用户名进行搜索。

```

<username-to-dn>
    <username-filter base-dn="dc=people,dc=harold,dc=example,dc=com"
recursive="false" attribute="sn" user-dn-attribute="dn" />
</username-to-dn>

```

可以在这里设置的属性是：

- **base-dn**: 开始搜索的上下文的标识名。

- **recursive** : 搜索是否将扩展到子上下文。默认值为 **false**。
- **attribute** : 根据提供的用户名尝试和匹配的用户条目的属性。默认值为 **uid**。
- **user-dn-attribute** : 为获得用户标识名而读取的属性。默认值为 **dn**。

advanced-filter

这个最后的选项指定了高级过滤器，和验证部分一样，这是使用自定义过滤器来定位用户标识名的机会。

```
<username-to-dn>
  <advanced-filter base-dn="dc=people,dc=harold,dc=example,dc=com"
recursive="false" filter="sAMAccountName={0}" user-dn-attribute="dn" />
</username-to-dn>
```

username-filter 示例里对应的属性的含义和默认值都是一样的，这里不再列出。新的属性是：

- **filter** : 用于搜索用户条目的过滤器，用户名将在占位符 **{0}** 里替换。



重要

XML 格式必须保持有效，定义过滤器来确保特殊字符（如 **&**）的正确格式。例如，对于 **&** 字符是 **&**。

组搜索

如上面所描述的，在搜索组成员资格信息时你可以使用两种风格。第一种是用户条目包含引用用户所属组的属性。第二种风格是组包含引用用户条目的属性。

当可以选择风格时，红帽推荐引用所用组的用户条目的配置。这是因为用这个方法时组信息可以通过读取已知标识名来加载而无需执行任何搜索。另外一个方法则要求额外的搜索来确定引用用户的组。

在描述配置之前，我们用一些 LDIF 例子来进行解释。

例 11.21. Principal to Group - LDIF 示例。

这个例子展示了用户 **TestUserOne**，它是 **GroupOne** 的成员，而 **GroupOne** 也是 **GroupFive** 的成员。通过属性 **memberOf** 展示了组成员资格，它被设置为用户（或组）所属的组的标识名。

这里没有展示的是用户可能具有多个 **memberOf** 的属性集，每个都对应该用户所属的组。

```
dn: uid=TestUserOne,ou=users,dc=principal-to-group,dc=example,dc=org
objectClass: extensibleObject
objectClass: top
objectClass: groupMember
objectClass: inetOrgPerson
objectClass: uidObject
objectClass: person
objectClass: organizationalPerson
cn: Test User One
sn: Test User One
uid: TestUserOne
```



```

distinguishedName: uid=TestUserOne,ou=users,dc=principal-to-
group,dc=example,dc=org
memberOf: uid=GroupOne,ou=groups,dc=principal-to-group,dc=example,dc=org
memberOf: uid=Slashy/Group,ou=groups,dc=principal-to-
group,dc=example,dc=org
userPassword::
e1NTSEF9WFpURzhLVjc4WVZBQUJNbEI3Ym96UVAva0RTNlFNWUpLOTdTMUE9PQ==

dn: uid=GroupOne,ou=groups,dc=principal-to-group,dc=example,dc=org
objectClass: extensibleObject
objectClass: top
objectClass: groupMember
objectClass: group
objectClass: uidObject
uid: GroupOne
distinguishedName: uid=GroupOne,ou=groups,dc=principal-to-
group,dc=example,dc=org
memberOf: uid=GroupFive,ou=subgroups,ou=groups,dc=principal-to-
group,dc=example,dc=org

dn: uid=GroupFive,ou=subgroups,ou=groups,dc=principal-to-
group,dc=example,dc=org
objectClass: extensibleObject
objectClass: top
objectClass: groupMember
objectClass: group
objectClass: uidObject
uid: GroupFive
distinguishedName: uid=GroupFive,ou=subgroups,ou=groups,dc=principal-to-
group,dc=example,dc=org

```

例 11.22. Group to Principal - LDIF 示例

这个例子展示了相同的用户 **TestUserOne**，它是 **GroupOne** 的成员（反之也是 **GroupFive** 的成员）- 然而在这个例子里，它是一个用于交叉引用的组到用户的属性 **uniqueMember**。

用于组成员资格交叉引用的属性可以重复，如果你查看 **GroupFive**，那里也有一个没有显示在这里的对其他用户 **TestUserFive** 的引用。

```

dn: uid=TestUserOne,ou=users,dc=group-to-principal,dc=example,dc=org
objectClass: top
objectClass: inetOrgPerson
objectClass: uidObject
objectClass: person
objectClass: organizationalPerson
cn: Test User One
sn: Test User One
uid: TestUserOne
userPassword::
e1NTSEF9SjR00TRDR1ltaHc1VVZQ0EJvbXhUYjl1dkFVd1lQTmRLSEdzaWc9PQ==

dn: uid=GroupOne,ou=groups,dc=group-to-principal,dc=example,dc=org
objectClass: top
objectClass: groupOfUniqueNames

```

```

objectClass: uidObject
cn: Group One
uid: GroupOne
uniqueMember: uid=TestUserOne,ou=users,dc=group-to-
principal,dc=example,dc=org

dn: uid=GroupFive,ou=subgroups,ou=groups,dc=group-to-
principal,dc=example,dc=org
objectClass: top
objectClass: groupOfUniqueNames
objectClass: uidObject
cn: Group Five
uid: GroupFive
uniqueMember: uid=TestUserFive,ou=users,dc=group-to-
principal,dc=example,dc=org
uniqueMember: uid=GroupOne,ou=groups,dc=group-to-
principal,dc=example,dc=org

```

通用组搜索

通过上面例子里展示的两种方法，我们知道首先需要定义两者共用的属性。

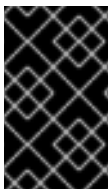
```

<group-search group-name="..." iterative="..." group-dn-attribute="..."
group-name-attribute="..." >
...
</group-search>

```

- **group-name**：这个属性用来指定用于作为用户所属组列表返回的组名的格式。这可以是组名的简单格式或者是组的标识名，如果标识名是必需的，那这个属性可以设置为 **DISTINGUISHED_NAME**。默认值是 **SIMPLE**。
- **iterative**：这个属性用来指定在确定用户所属的组之后是否应该根据组所属的组迭代地进行搜索。如果启用了迭代搜索，那么我们将继续搜索，直到到达不是成员的组或检测到循环。它的默认值为 **false**。

循环的组成员并不是一个问题。我们保存了每次搜索的记录来防止已搜索过的组被再次搜索。



重要

为了迭代式搜索能正常进行，组条目需要和用户条目一致，也就是使用相同的方法来确定用户所属的组和确定组所属组。但对于组到组的成员关系，如果用于交叉引用的属性或者引用方向有改动，那这就无法实现了。

- **group-dn-attribute**：组条目上的哪个属性是其标识名。默认为 **dn**。
- **group-name-attribute**：组条目上的哪个属性是其简单名。默认为 **uid**。

例 11.23. Principal to Group 配置示例

基于上面的 LDIF 例子，下面是一个循环加载用户组的配置示例，用来进行交叉引用的属性是用户的 **memberOf**。

```
<authorization>
```

```

<ldap connection="LocalLdap">
  <username-to-dn>
    <username-filter base-dn="ou=users,dc=principal-to-
group,dc=example,dc=org" recursive="false" attribute="uid" user-dn-
attribute="dn" />
  </username-to-dn>
  <group-search group-name="SIMPLE" iterative="true" group-dn-
attribute="dn" group-name-attribute="uid">
    <principal-to-group group-attribute="memberOf" />
  </group-search>
</ldap>
</authorization>

```

这个配置最重要的方面是已添加了带有单个属性的 **principal-to-group** 元素。

- **group-attribute** :

例 11.24. Group to Principal 配置示例

这个例子展示了一个对上面的 group to principal LDIF 例子的迭代搜索。

```

<authorization>
  <ldap connection="LocalLdap">
    <username-to-dn>
      <username-filter base-dn="ou=users,dc=group-to-
principal,dc=example,dc=org" recursive="false" attribute="uid" user-dn-
attribute="dn" />
    </username-to-dn>
    <group-search group-name="SIMPLE" iterative="true" group-dn-
attribute="dn" group-name-attribute="uid">
      <group-to-principal base-dn="ou=groups,dc=group-to-
principal,dc=example,dc=org" recursive="true" search-
by="DISTINGUISHED_NAME">
        <membership-filter principal-attribute="uniqueMember"
/>
      </group-to-principal>
    </group-search>
  </ldap>
</authorization>

```

在这里添加了 **group-to-principal** 元素，这个元素用来定义引用用户条目的组搜索如何只执行，它将设置下列属性：

- **base-dn**: 开始搜索的上下文的标识名。
- **recursive** : 是否搜索子上下文。默认为 **false**。
- **search-by** : 在搜索里使用的角色名称的格式。有效值是 **SIMPLE** 和 **DISTINGUISHED_NAME**。默认值为 **DISTINGUISHED_NAME**。

在 **group-to-principal** 元素里有一个定义交叉引用的 **membership-filter** 元素。

- **principal-attribute** : 引用用户条目的组条目上的属性名称。默认值为 **member**。

[提交 bug 报告](#)

11.9.9.8. 关于带作用域的角色

带作用域的角色 (**Scoped Role**) 是用户定义的角色，它赋予标准角色中一个角色的权限，但只用于一个或多个指定的服务器组或主机。带作用域的角色允许按需要赋予管理用户限于某些服务器组或主机的权限。

带作用域的角色可以由具有 **Administrator** 或 **SuperUser** 角色的用户来创建。

它们是由 5 个特征定义的：

1. 唯一的名字。
2. 基于那些标准角色。
3. 是否应用在服务器组或主机上。
4. 所限的服务器组或主机的列表。
5. 是否自动包括所有用户。默认为 **false**。

创建之后，带作用域的角色可以和标准角色一样分配给用户和组。

创建带作用域的角色并不意味着要定义新的权限。带作用域的角色只能用来在有限的作用域里应用现存角色的权限。例如，你可以根据限于单个服务器组的 **Deployer** 角色创建带作用域的角色。

角色只可以限于两个作用域，主机和服务器组。

作用域为主机的角色

作用域为主机的角色将它的权限限制到一个或多个主机。这意味着为相关的 **/host=*/** 资源树提供了访问权限但其他主机所专有的资源是隐藏的。

作用域为服务器组的角色

作用域为服务器组的角色限制了该角色的权限到一个或多个服务器组。此外，角色权限将应用于和指定的服务器组相关联的配置集、套接字绑定组、服务器配置和服务器资源。和这个服务器组逻辑上不关联的资源的任何子资源对于用户来说都是不可见的。

作用域为主机和服务器组的角色都具有受管域配置的其他部分的 **Monitor** 角色的权限。

[提交 bug 报告](#)

11.9.9.9. 创建带作用域的角色

带作用域的角色 (**Scoped Role**) 是用户定义的角色，它赋予标准角色中一个角色的权限，但只用于一个或多个指定的服务器组或主机。本节将展示如何创建带作用域的角色。

只有具有 **SuperUser** 或 **Administrator** 角色的用户才能执行这些配置。

在管理控制台里可通过下列步骤来配置带作用域的角色：

1. 登录到管理控制台
2. 点击 **Administration** 标签页
3. 展开 **Access Control** 菜单并选择 **Role Assignment**。
4. 选择 **ROLES** 标签页，然后点击里面的 **Scoped Roles** 标签页。

管理控制台的 **Scoped Roles** 部分由两个主要区域组成，包含当前配置的带作用域的角色表，以及显示表里当前选择的角色细节的 **Selection** 面板。

下面的过程展示了如何配置带作用域的角色。

过程 11.29. 添加新的带作用域的角色

1. 登录到管理控制台
2. 进入 **Roles** 标签页的 **Scoped Roles** 区域。
3. 点击 **Add**。**Add Scoped Role** 对话框将会出现。
4. 指定下列细节：
 - **Name**，新的带作用域的角色名称。
 - **Base Role**，这个角色的权限所基于的角色。
 - **Type**，这个角色是否将被限于主机或服务器组。
 - **Scope**，这个角色所限于的主机或服务器组列表。可以选择多重条目。
 - **Include All**，这个角色是否自动包含所有的用户。默认为 no。
5. 点击 **Save** 按钮，对话框将关闭且新创建的角色将出现在表里。

过程 11.30. 编辑带作用域的角色

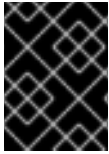
1. 登录到管理控制台
2. 进入 **Roles** 标签页的 **Scoped Roles** 区域。
3. 点击你要编辑的带作用域的角色。这个角色的细节将出现在表下面的 **Selection** 面板上。
4. 点击 **Selection** 面板里的 **Edit** 按钮。**Selection** 面板将进入编辑模式。
5. 更新你要修改的细节并点击 **Save** 按钮。**Selection** 面板将回到之前的状态。**Selection** 面板和表都会显示最近更新的细节。

过程 11.31. 查看 Scoped Role 成员

1. 登录到管理控制台
2. 进入 **Roles** 标签页的 **Scoped Roles** 区域。
3. 点击表里你要查看的 **Members** 的带作用域的角色，然后点击 **Members** 按钮。**Members of role** 对话框将出现。它将显示这个角色包含或排斥的用户和组。

4. 当你已经完成查看这些信息后请点击 **Done** 按钮。

过程 11.32. 删除带作用域的角色



重要

如果用户或组被分配至某个 **Scoped Role**，这个角色就不能被删除。你需要先删除角色分配。

1. 登录到管理控制台
2. 进入 **Roles** 标签页的 **Scoped Roles** 区域。
3. 选择表里要删除的带作用域的角色。
4. 点击 **Remove** 按钮。**Remove Scoped Role** 对话框将会出现。
5. 点击 **Confirm** 按钮。对话框将关闭且角色会被删除。

[提交 bug 报告](#)

11.9.10. 配置约束

11.9.10.1. 配置 Sensitivity 约束

每个敏感性约束（**Sensitivity Constraint**）都定义了被认为是“敏感的”的资源的集合。敏感的资源通常是某种机密的内容，如密码，或者对服务器的操作有着重大影响的东西，如网络、**JVM** 配置或系统属性。访问控制系统本身也被视作敏感的资源。资源敏感性限制了哪些角色可以读、写或寻址专门的资源。

应用程序约束的配置位于管理 API 里的 `/core-service=management/access=authorization/constraint=sensitivity-classification`。

在管理模型内部，每个 **Sensitivity** 约束都被标识为**类别（classification）**。这些类别被分组为**类型（types）**。有 39 种类别被分组成 13 个类型。

要配置 **Sensitivity** 约束，请使用 **write-attribute** 操作来设置 **configured-requires-read**、**configured-requires-write** 或 **configured-requires-addressable** 属性。要使该类型的操作成为敏感的，请将其设置为 **false**。在默认情况下，它们不会被设置而使用 **default-requires-read**、**default-requires-write** 和 **default-requires-addressable** 的值。一旦配置的属性被设置，它将替代默认值。默认值不能被修改。

例 11.25. 使读取系统属性成为敏感性操作

```
[domain@localhost:9999 /] cd /core-
service=management/access=authorization/constraint=sensitivity-
classification/type=core/classification=system-property
[domain@localhost:9999 classification=system-property] :write-
attribute(name=configured-requires-read, value=true)
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
```

```

        "server-one" => {"response" => {"outcome" => "success"}},
        "server-two" => {"response" => {"outcome" => "success"}}
    }
}
[domain@localhost:9999 classification=system-property] :read-resource
{
    "outcome" => "success",
    "result" => {
        "configured-requires-addressable" => undefined,
        "configured-requires-read" => true,
        "configured-requires-write" => undefined,
        "default-requires-addressable" => false,
        "default-requires-read" => false,
        "default-requires-write" => true,
        "applies-to" => {
            "/host=master/system-property=" => undefined,
            "/host=master/core-service=platform-mbean/type=runtime" =>
undefined,
            "/server-group=*/system-property=" => undefined,
            "/host=master/server-config=*/system-property=" =>
undefined,
            "/host=master" => undefined,
            "/system-property=" => undefined,
            "/" => undefined
        }
    }
}
[domain@localhost:9999 classification=system-property]

```

表 11.6 “Sensitivity 约束配置结果”总结了哪些角色能够执行哪些操作所取决于的配置。

表 11.6. Sensitivity 约束配置结果

Value	requires-read	requires-write	requires-addressable
true	读操作是敏感的。 只有 Auditor 、 Administrator 和 SuperUser 有读的权限。	写操作是敏感的。 只有 Administrator 和 SuperUser 有写的权限。	寻址 (Addressing) 是敏感的。 只有 Auditor 、 Administrator 和 SuperUser 可以寻址。
false	读操作是非敏感的。 任何管理用户都可以读。	写操作是非敏感的。 只有 Maintainer 、 Administrator 和 SuperUser 可以写。如果这个约束是一个应用程序资源的话， Deployers 也可以写。	寻址 (Addressing) 是非敏感的。 任何管理用户都可以寻址。

[提交 bug 报告](#)

11.9.10.2. 配置应用程序资源约束

每个应用程序资源约束都定义了一系列通常和应用程序及服务部署相关联的资源、属性和操作。当启用了应用程序资源约束时，具有 **Deployer** 角色的管理用户会被赋予访问资源的权限。

应用程序约束配置位于 **/core-**

service=management/access=authorization/constraint=application-classification/ 里的管理模型。

在管理模型内部，每个应用程序资源约束都被标识为**类别（classification）**。这些类别被分组为**类型（types）**。有 14 种类别被分组成 8 个类型。每个类别都有一个 **applies-to** 元素，它是一个类别配置应用的资源路径模式的列表。

在默认情况下，唯一启用的应用程序资源类别是 **core**。Core 包含部署、部署重叠和部署操作。

要启用应用程序资源，请使用 **write-attribute** 操作来设置 **configured-application attribute** 为 **true**。要禁用应用程序资源，请设置这个属性为 **false**。在默认情况下，这些属性没被设置且使用了 **default-application** 属性的值。这个默认值不能被修改。

例 11.26. 启用 logger-profile 应用程序资源归类

```
[domain@localhost:9999 /] cd /core-
service=management/access=authorization/constraint=application-
classification/type=logging/classification=logging-profile
[domain@localhost:9999 classification=logging-profile] :write-
attribute(name=configured-application, value=true)
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {"outcome" => "success"}},
    "server-two" => {"response" => {"outcome" => "success"}}
  }}}
}
[domain@localhost:9999 classification=logging-profile] :read-resource
{
  "outcome" => "success",
  "result" => {
    "configured-application" => true,
    "default-application" => false,
    "applies-to" => {"/profile=*/subsystem=logging/logging-
profile=*" => undefined}
  }
}
[domain@localhost:9999 classification=logging-profile]
```

重要

应用程序资源约束应用域所有匹配它的配置的资源。例如，只将 **Deployer** 用户权限赋予一个数据源资源而不是另外一个是不可能的。如果需要这种级别的分离，那么我们推荐在其他的服务器组里配置资源并为每个组创建不同作用域的 **Deployer** 角色。

11.9.10.3. 配置 Vault 表达式约束

在默认情况下，读和写 Vault 表达式是敏感操作。配置 Vault 表达式约束允许你设置这两个操作为非敏感的。修改这个约束允许更多的角色读和写 Vault 表达式。

Vault 表达式约束的配置位于 `/core-`

`service=management/access=authorization/constraint=vault-expression` 上的管理模型里。

要配置 Vault 表达式，请使用 `write-attribute` 操作来设置 `configured-requires-write` 和 `configured-requires-read` 的属性为 `true` 或 `false`。在默认情况下，它们不会被设置而使用 `default-requires-read` 和 `default-requires-write` 的值。默认值不能被修改。

例 11.27. 使写入 Vault 表示成为非敏感操作

```
[domain@localhost:9999 /] cd /core-
service=management/access=authorization/constraint=vault-expression
[domain@localhost:9999 constraint=vault-expression] :write-
attribute(name=configured-requires-write, value=false)
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {"outcome" => "success"}}},
    "server-two" => {"response" => {"outcome" => "success"}}
  }}}
}
[domain@localhost:9999 constraint=vault-expression] :read-resource
{
  "outcome" => "success",
  "result" => {
    "configured-requires-read" => undefined,
    "configured-requires-write" => false,
    "default-requires-read" => true,
    "default-requires-write" => true
  }
}
[domain@localhost:9999 constraint=vault-expression]
```

表 11.7 “Vault 表达式约束配置结果”总结了哪些角色能够读取和写入 Vault 表达式所取决于的配置。

表 11.7. Vault 表达式约束配置结果

Value	requires-read	requires-write
true	读操作是敏感的。 只有 Auditor 、 Administrator 和 SuperUser 有读的权限。	写操作是敏感的。 只有 Administrator 和 SuperUser 有写的权限。

Value	requires-read	requires-write
false	<p>读操作是不敏感的。</p> <p>所有管理用户都可以读。</p>	<p>写操作是不敏感的。</p> <p>Monitor、Administrator 和 SuperUser 有写权限。如果这个约束是一个应用程序资源的话，Deployers 也可以写。</p>

[提交 bug 报告](#)

11.9.11. 约束引用

11.9.11.1. 应用程序资源约束引用

类型：**core**

类别：**deployment-overlay**

- 默认值：**true**
- PATH: /deployment-overlay=*
- PATH: /deployment=*
- PATH: /

操作：

upload-deployment-stream, full-replace-deployment, upload-deployment-url, upload-deployment-bytes

类型：**datasources**

类别：**datasource**

- 默认值：**false**
- PATH: /deployment=*/subdeployment=*/subsystem=datasources/data-source=*
- PATH: /subsystem=datasources/data-source=*
- PATH: /subsystem=datasources/data-source=ExampleDS
- PATH: /deployment=*/subsystem=datasources/data-source=*

Classification: **jdbc-driver**

- 默认值：**false**
- PATH: /subsystem=datasources/jdbc-driver=*

类别 : xa-data-source

- 默认值 : false
- PATH: /subsystem=datasources/xa-data-source=*
- PATH: /deployment=*/subsystem=datasources/xa-data-source=*
- PATH: /deployment=*/subdeployment=*/subsystem=datasources/xa-data-source=*

类型 : logging**类别 : logger**

- 默认值 : false
- PATH: /subsystem=logging/logger=*
- PATH: /subsystem=logging/logging-profile=*/logger=*

类别 : logging-profile

- 默认值 : false
- PATH: /subsystem=logging/logging-profile=*

类型 : mail**类别 : mail-session**

- 默认值 : false
- PATH: /subsystem=mail/mail-session=*

类型 : naming**类别 : binding**

- 默认值 : false
- PATH: /subsystem=naming/binding=*

类型 : resource-adapters**类别 : resource-adapters**

- 默认值 : false
- PATH: /subsystem=resource-adapters/resource-adapter=*

类型 : security

类别 : **security-domain**

- 默认值 : **false**
- PATH: /subsystem=security/security-domain=*

[提交 bug 报告](#)

11.9.11.2. 安全约束引用

类型 : **core**

类别 : **access-control**

- requires-addressable: **true**
- requires-read: **true**
- requires-write: **true**
- PATH: /core-service=management/access=authorization
- PATH: /subsystem=jmx ATTRIBUTE: non-core-mbean-sensitivity

类别 : **credential**

- requires-addressable: **false**
- requires-read: **true**
- requires-write: **true**
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: username , password
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: username , password
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: user-name, recovery-username, password, recovery-password
- PATH: /subsystem=mail/mail-session=*/custom=* ATTRIBUTE: username, password
- PATH: /subsystem=datasources/data-source=*" ATTRIBUTE: user-name, password
- PATH: /subsystem=remoting/remote-outbound-connection=*" ATTRIBUTE: username
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: username, password
- PATH: /subsystem=web/connector=*/configuration=ssl ATTRIBUTE: key-alias, password
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=*" ATTRIBUTE: recovery-username, recovery-password

类别 : **domain-controller**

- requires-addressable: **false**

- requires-read: false
- requires-write: true

类别 : **domain-names**

- requires-addressable: false
- requires-read: false
- requires-write: true

类别 : **extensions**

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /extension=*

类别 : **jvm**

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=platform-mbean/type=runtime ATTRIBUTE: input-arguments, boot-class-path, class-path, boot-class-path-supported, library-path

类别 : **management-interfaces**

- requires-addressable: false
- requires-read: false
- requires-write: true
- /core-service=management/management-interface=native-interface
- /core-service=management/management-interface=http-interface

类别 : **module-loading**

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=module-loading

类别 : **patching**

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=patching/addon=*
- PATH: /core-service=patching/layer=*
- PATH: /core-service=patching

类别 : read-whole-config

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: / OPERATION: read-config-as-xml

类别 : security-domain

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /subsystem=security/security-domain=*

类别 : security-domain-ref

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: security-domain
- PATH: /subsystem=datasources/data-source=* ATTRIBUTE: security-domain
- PATH: /subsystem=ejb3 ATTRIBUTE: default-security-domain
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=*
ATTRIBUTE: security-domain, recovery-security-domain, security-application, security-domain-and-application

类别 : security-realm

- requires-addressable: true
- requires-read: true
- requires-write: true

- PATH: /core-service=management/security-realm=*

类别 : **security-realm-ref**

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: security-realm
- PATH: /core-service=management/management-interface=native-interface ATTRIBUTE: security-realm
- PATH: /core-service=management/management-interface=http-interface ATTRIBUTE: security-realm
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: security-realm

类别 : **security-vault**

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=vault

类别 : **service-container**

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=service-container

类别 : **snapshots**

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: / ATTRIBUTE: take-snapshot, list-snapshots, delete-snapshot

类别 : **socket-binding-ref**

- requires-addressable: false
- requires-read: false

- requires-write: false
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: socket-binding
- PATH: /subsystem=web/connector=* ATTRIBUTE: socket-binding
- PATH: /subsystem=remoting/local-outbound-connection=* ATTRIBUTE: outbound-socket-binding-ref
- PATH: /socket-binding-group=*/local-destination-outbound-socket-binding=* ATTRIBUTE: socket-binding-ref
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=transactions ATTRIBUTE: process-id-socket-binding, status-socket-binding, socket-binding

类别 : **socket-config**

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /interface=* OPERATION: resolve-internet-address
- PATH: /core-service=management/management-interface=native-interface ATTRIBUTE: port, interface, socket-binding
- PATH: /socket-binding-group=*
- PATH: /core-service=management/management-interface=http-interface ATTRIBUTE: port, secure-port, interface, secure-socket-binding, socket-binding
- PATH: / OPERATION: resolve-internet-address
- PATH: /subsystem=transactions ATTRIBUTE: process-id-socket-max-ports

类别 : **system-property**

- requires-addressable: false
- requires-read: false
- requires-write: true

- PATH: /core-service=platform-mbean/type=runtime ATTRIBUTE: system-properties
- PATH: /system-property=*
- PATH: / OPERATION: resolve-expression

类型 : **datasources**

类别 : **data-source-security**

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: user-name, security-domain, password
- PATH: /subsystem=datasources/data-source=* ATTRIBUTE: user-name, security-domain, password

类型 : **jdr**

类别 : **jdr**

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /subsystem=jdr OPERATION: generate-jdr-report

类型 : **jmx**

类别 : **jmx**

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /subsystem=jmx

类型 : **mail**

类别 : **mail-server-security**

- requires-addressable: false
- requires-read: false

- requires-write: true
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=*/custom=* ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: username, tls, ssl, password

类型 : **naming**

类别 : **jndi-view**

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=naming OPERATION: jndi-view

类别 : **naming-binding**

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=naming/binding=*

类型 : **remoting**

类别 : **remoting-security**

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: authentication-provider, security-realm
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: username, security-realm
- PATH: /subsystem=remoting/connector=*/security=sasl

类型 : **resource-adapters**

Classification: **resource-adapter-security**

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=*
- ATTRIBUTE: security-domain, recovery-username, recovery-security-domain, security-application, security-domain-and-application, recovery-password

类型 : **security**

类别 : **misc-security**

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=security ATTRIBUTE: deep-copy-subject-mode

类型 : **web**

类别 : **web-access-log**

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=web/virtual-server=*/configuration=access-log

类别 : **web-connector**

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=web/connector=*

类别 : **web-ssl**

- requires-addressable: false
- requires-read: true
- requires-write: true

- PATH: /subsystem=web/connector=*/configuration=ssl

类别：web-sso

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=web/virtual-server=*/configuration=sso

类别：web-valve

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=web/valve=*

[提交 bug 报告](#)

11.10. 网络安全性

11.10.1. 保护管理接口

概述

在测试环境里，我们通常在由管理控制台、管理 CLI 和其他 API 实现组成的管理接口上不设置安全层来运行 JBoss EAP 6。这可以允许快速的开发和配置修改。

此外，`silent` 验证模式也是默认的，这允许主机上的本地客户可以连接管理 CLI 而无需用户名和密码。这对于本地用户和管理 CLI 脚本是很方便的，但它可以根据需要禁用。其步骤请参考 [第 11.8.6 节“从默认的安全区删除无提示验证”](#)。

当你开始测试并准备移至产品环境时，至少通过下列方式保护管理接口是极其重要的：

- [第 11.10.2 节“指定 JBoss EAP 6 使用的网络接口”](#)
- [第 11.10.4 节“配置和 JBoss EAP 6 一起使用的网络防火墙”](#)

[提交 bug 报告](#)

11.10.2. 指定 JBoss EAP 6 使用的网络接口

介绍

隔离服务以使它们只被需要它们的客户访问，这样可以增强网络的安全性。JBoss EAP 在其默认配置里包含两个接口，两者都绑定到 IP 地址 `127.0.0.1` 或 `localhost`。其中一个被称为 `management`，它被管理控制台、CLI 和 API 使用。另外一个被称为 `public`，被用来部署应用程序。这些接口并不特殊或有什么含义，只是作为起点来提供。

management 接口默认使用端口 **9990** 和 **9999**，而 **public** 接口则使用端口 **8080**，或者 **8443**（如果您使用 HTTPS）。

你修改管理接口或公共接口的 IP 地址，或者两者都修改。



警告

如果你将管理接口开放给远程主机可以访问的其他网络接口，请注意潜在的安全问题。多数情况下，我们不建议为管理接口提供远程访问。

1. 停止 JBoss EAP 6 服务器。

以合适的方式对操作系统发送中断信号来停止 JBoss EAP 6。如果你以前台应用程序的方式运行 JBoss EAP 6，那通常的做法是按 **Ctrl+C** 键。

2. 重启 JBoss EAP 6，指定绑定地址。

使用 **-b** 命令行选项在特定接口上启动 JBoss EAP 6。

例 11.28. 指定公共接口。

```
EAP_HOME/bin/domain.sh -b 10.1.1.1
```

例 11.29. 指定管理接口。

```
EAP_HOME/bin/domain.sh -bmanagement=10.1.1.1
```

例 11.30. 为每个接口指定不同的地址。

```
EAP_HOME/bin/domain.sh -bmanagement=127.0.0.1 -b 10.1.1.1
```

例 11.31. 绑定公共接口到所有的网络接口上。

```
EAP_HOME/bin/domain.sh -b 0.0.0.0
```

您也可以直接编辑你的 XML 配置文件以修改默认的绑定地址。然而，如果您这么做，您将不能使用 **-b** 命令行选项在运行时指定 IP 地址，所以我们不推荐这么做。如果确实要这么做，请在编辑 XML 文件前完全停止 JBoss EAP 6。

[提交 bug 报告](#)

11.10.3. JBoss EAP 6 使用的网络端口

JBoss EAP 6 的默认配置使用的端口取决于下列因素：

- 你的服务器组是否使用了默认的套接字绑定组，或者自定义的套接字绑定组。
- 单独部署的要求。



注意

你可以配置一个数字的端口偏移量，以减缓当在同一个服务服务器上运行多个服务器时端口冲突。如果你的服务器使用了数字的端口偏移量，在它的服务器组的套接字绑定组的默认端口上添加偏移量。例如，如果套接字绑定组的 HTTP 端口是 **8080**，而你的服务器使用了端口偏移量为 **100**，那么它的 HTTP 端口是 **8180**。

除非额外指定，这个端口将使用 TCP 协议。

默认的套接字绑定组

- **full-ha-sockets**
- **full-sockets**
- **ha-sockets**
- **standard-sockets**

表 11.8. 默认的套接字绑定组的引用

名称	端口	多点传送 端口	描述	full-ha- sockets	full- sockets	ha- socket	standar d- socket
ajp	8009		Apache JServ 协议， 用于 HTTP 群集和负 载平衡。	支持	支持	支持	支持
http	8080		用于已部署应用程序 的默认端口。	支持	支持	支持	支持
https	8443		已部署的应用程序和 客户间的用 SSL 加密 的连接。	支持	支持	支持	支持
jacorb	3528		用于 JTS 事务的 CORBA 服务和其他依 赖于 ORB 的服务。	支持	支持	不支持	不支持
jacorb -ssl	3529		SSL 加密的 CORBA 服务。	支持	支持	不支持	不支持
jgroup s- diagno stics		7500	多点传送。用于 HA 群集里的 Peer 发现。 不能使用管理界面进 行配置。	支持	不支持	支持	不支持

名称	端口	多点传送 端口	描述	full-ha- sockets	full- sockets	ha- socket	standar d- socket
jgroup s- mping		45700	多点传送。用于在 HA 群集里发现初始成员资格。	支持	不支持	支持	不支持
jgroup s-tcp	7600		HA 群集里使用 TCP 的多点传送 Peer 发现。	支持	不支持	支持	不支持
jgroup s-tcp- fd	57600		用于 TCP 上的 HA 失败检测。	支持	不支持	支持	不支持
jgroup s-udp	55200	45688	HA 群集里使用 UDP 的多点传送 Peer 发现。	支持	不支持	支持	不支持
jgroup s-udp- fd	54200		用于 UDP 上的 HA 失败检测。	支持	不支持	支持	不支持
messaging	5445		JMS 服务。	支持	支持	不支持	不支持
messaging- group			被 HornetQ JMS 广播和发现组引用。	支持	支持	不支持	不支持
messaging- throughput	5455		JMS remoting 所使用的。	支持	支持	不支持	不支持
mod_cluster		23364	用于 JBoss EAP 6 和 HTTP 加载平衡器之间通讯的多点传送端口。	支持	不支持	支持	不支持
osgi- http	8090		由使用 OSGi 子系统的内部组件使用。不能通过管理界面进行配置。	支持	支持	支持	支持
remoting	4447		用于远程 EJB 调用。	支持	支持	支持	支持

名称	端口	多点传送 端口	描述	full-ha- sockets	full- sockets	ha- socket	standar d- socket
txn- recove ry- enviro nment	4712		JTA 事务恢复管理 者。	支持	支持	支持	支持
txn- status - manage r	4713		JTA / JTS 事务管理 者。	支持	支持	支持	支持

管理端口

除了套接字绑定组，每个主机控制台都打开另外两个端口用于管理：

- **9990** - Web 管理控制台的端口
- **9999** - 管理控制台和 API 使用的端口

此外，如果管理控制台启用了 HTTPS，那么 **9443** 将作为默认端口打开。

[提交 bug 报告](#)

11.10.4. 配置和 JBoss EAP 6 一起使用的网络防火墙

概述

多数产品环境都使用防火墙作为总体网络安全策略的一部分。如果你需要多个 EAP 服务器来与其他 EAP 服务器或外部服务（如 Web 服务器或数据库）来通讯，你的防火墙必须考虑这一点。管理良好的防火墙只会打开操作所需的端口，它会限制对某些 IP 地址、子网和网络协议的端口的访问。

不过，关于防火墙的完整讨论超出了本文档的范畴。

必须具备的条件

- 确定你要打开的端口。
- 对防火墙软件的理解是必需的。在红帽企业版 Linux 6 里，这个过程会使用 **system-config-firewall** 命令。微软的 Windows 服务器包括了内置的防火墙，以及几个可用于任何平台的第三方的防火墙解决方案。在微软的 Windows 服务器上，您可以使用 PowerShell 来配置防火墙。

假设

这个过程配置防火墙是基于如下假设的：

- 操作系统为红帽企业版 Linux 6。
- JBoss EAP 6 运行在主机 **10.1.1.2** 上。或者，EAP 服务器具有自己的防火墙。
- 网络防火墙服务器运行在主机 **10.1.1.1** 的接口 **eth0** 上，且具有外部的接口 **eth1**。

- 你想把端口 **5445**（JMS 使用的端口）上的流量转发到 JBoss EAP 6。网络防火墙应该不允许其他流量通过。

过程 11.33. 管理和 JBoss EAP 6 一起使用的网络防火墙

1. 登录到管理控制台。

登录到管理控制台。在默认情况下，它运行在 <http://localhost:9990/console/> 上。

2. 确定套接字绑定组使用的套接字绑定。

- 点击管理控制台顶部的 **Configuration** 标签。
- 展开 **General Configuration** 菜单，选择 **Socket Binding**。
- Socket Binding Declarations** 屏幕将会出现。首先将显示 **standard-sockets** 组。你可以从右边的组合框里选择不同的组。



注意

如果你使用的是独立服务器，它只有一个套接字绑定组。

套接字名称和端口将会出现，每页有 8 个值。你可以使用表下面的箭头来进行浏览。

3. 确定你要打开的端口。

根据特定端口的功能以及系统环境的要求，你可能需要在防火墙上打开某些端口。

4. 配置防火墙将通讯转发到 JBoss EAP 6。

执行这些步骤来配置网络防火墙以允许指定端口上的通讯。

- 登录到防火墙主机并以根用户身份打开命令行提示。
- 执行 **system-config-firewall** 来启动防火墙配置工具。GUI 或命令行工具根据你登录到防火墙系统的方式启动。这个任务假设你通过 SSH 登录并使用了命令行接口。
- 使用 **TAB** 键切换到 **Customize** 按钮，然后按 **ENTER** 键。**Trusted Services** 屏幕将会出现。
- 不要修改任何值，使用 **TAB** 键切换到 **Forward** 按钮，然后按 **ENTER** 键进入下一屏幕。**Other Ports** 屏幕将出现。
- 使用 **TAB** 键切换到 **<Add>** 按钮，然后按 **ENTER** 键。**Port and Protocol** 屏幕将会出现。
- 在 **Port / Port Range** 字段里输入 **5445**，然后使用 **TAB** 键切换到 **Protocol** 字段，并输入 **tcp**。使用 **TAB** 键切换到 **OK** 按钮，然后按 **ENTER** 键。
- 使用 **TAB** 键切换到 **Forward** 按钮，直至你到达 **Port Forwarding** 屏幕。
- 使用 **TAB** 键切换到 **<Add>** 按钮，然后按 **ENTER** 键。
- 输入下列值来设立端口 **5445** 的端口转发。
 - 源接口：**eth1**
 - 协议：**tcp**

- 端口/端口范围：5445
- 目的地 IP 地址：10.1.1.2
- 端口/端口范围：5445

使用 **TAB** 键切换到 **OK** 按钮，然后按 **ENTER**。

j. 使用 **TAB** 键切换到 **Close** 按钮，然后按 **ENTER**。

k. 使用 **TAB** 键切换到 **OK** 按钮，然后按 **ENTER**。要应用这些改动，阅读警告提示并点击 **Yes**。

5. 在你的 JBoss EAP 6 主机上配置防火墙。

一些机构选择在 JBoss EAP 6 服务器上配置防火墙，并关闭非操作必需的端口。请参考第 11.10.3 节“JBoss EAP 6 使用的网络端口”并确定要打开的端口，然后关闭其他端口。红帽企业版 6 的默认配置关闭所有的端口，除了 22（用于 SSH）和 5353（用于多点传送 DNS）。当你在配置端口的时候，请确保你可以从物理上访问服务器，这样就不会不经意地将自己关在外面。

结果

配置了防火墙，它会按照你在防火墙配置里指定的将通讯转发到内部的 JBoss EAP 6 服务器。如果你选择在服务器上启用防火墙，除了需要运行应用程序的端口，所有的端口将被关闭。

过程 11.34. 在 Microsoft Windows 上用 PowerShell 配置防火墙

1. 为了进行调试，关闭防火墙以确定当前的网络是否和防火墙配置相关。

```
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-command "NetSh Advfirewall set allprofiles state off"'
```

2. 允许 UDP 连接端口 23364。例如：

```
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-command "NetSh Advfirewall firewall add rule name="UDP Port 23364" dir=in action=allow protocol=UDP localport=23364"'\nStart-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-command "NetSh Advfirewall firewall add rule name="UDP Port 23364" dir=out action=allow protocol=UDP localport=23364"'
```

过程 11.35. 配置 Red Hat 企业版 Linux 7 上的防火墙以允许 mod_cluster Advertising

- 为了在 Red Hat 企业版 Linux 7 里使用 mod_cluster advertising，您必须启用防火墙里的 UDP 端口：

```
firewall-cmd --permanent --zone=public --add-port=23364/udp
```



注意

224.0.1.105:23364 是 mod_cluster 负载均衡器广告 UDP 多点传送地默认地址和端口。

[提交 bug 报告](#)

11.11. JAVA 安全性管理者

11.11.1. 关于 Java 安全性管理者

Java 安全性管理者 (Java Security Manager)

Java 安全性管理者是一个管理 Java 虚拟机 (JVM) sandbox 外部边界的类，它控制代码在 JVM 里执行时如何和外部的资源交互。当 Java 安全性管理者被激活时，Java API 在执行许多有潜在风险的操作之前会检查 Java 安全性管理者以获得批准。

Java 安全性管理者使用一个安全策略来确定是否允许或拒绝给定的动作。

[提交 bug 报告](#)

11.11.2. 在 Java 安全管理者里运行 JBoss EAP 6

要指定一个 Java 安全管理者策略，你需要编辑在引导过程中传递给域或服务器实例的 Java 选项。出于这个原因，你不能将参数作为选项传递给 `domain.sh` 或 `standalone.sh` 脚本。下面的过程会指引你配置实例以在 Java 安全管理者里运行 JBoss EAP。

前提条件

- 在执行下列过程之前，你需要用 JDK 里包含的 `policytool` 编写一个安全策略。这个过程假定你的策略位于 `EAP_HOME/bin/server.policy`。您也可以使用任何文本编辑器手工编写安全策略并将其保存为 `EAP_HOME/bin/server.policy`。
- 在编辑配置文件之前，域或独立服务器必须完全停止。

如果你有域成员跨越多个系统，请对每个域里的物理主机或实例执行下列过程。

过程 11.36. 为 JBoss EAP 6 配置安全管理者

1. 打开配置文件。

打开配置文件进行编辑。根据你使用的是域还是独立服务器，这个文件位于两个位置中的一个。它不是用来启动服务器或域的可执行文件。

◦ 受管域

- 对于 Linux : `EAP_HOME/bin/domain.conf`
- 对于 Windows : `EAP_HOME\bin\domain.conf.bat`

◦ 独立服务器

- 对于 Linux : `EAP_HOME/bin/standalone.conf`
- 对于 Windows : `EAP_HOME\bin\standalone.conf.bat`

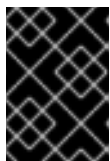
2. 在文件里添加 Java 选项。

为了确保使用 Java 选项，请在代码块里添加以下列行开始的内容：

```
if [ "x$JAVA_OPTS" = "x" ]; then
```

您可以修改 `-Djava.security.policy` 值来指定安全策略的准确位置。它只应该有一行，且没有换行符。设置 `-Djava.security.policy` 属性时使用 `==` 指定这个安全管理者将只使用指

定的策略文件。使用 `=` 指定安全管理者将使用指定的策略以及 `JAVA_HOME/lib/security/java.security` 的 `policy.url` 部分里的策略。



重要

JBoss EAP 从 6.2.2 开始要求将系统属性 `jboss.modules.policy-permissions` 设置为 `true`。

例 11.32. domain.conf

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.manager -
Djava.security.policy==${PWD}/server.policy -
Djboss.home.dir=/path/to/EAP_HOME -Djboss.modules.policy-
permissions=true"
```

例 11.33. domain.conf.bat

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.security.manager -
Djava.security.policy==\path\to\server.policy -
Djboss.home.dir=\path\to\EAP_HOME -Djboss.modules.policy-
permissions=true"
```

例 11.34. standalone.conf

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.manager -
Djava.security.policy==${PWD}/server.policy -
Djboss.home.dir=${JBoss_HOME} -Djboss.modules.policy-
permissions=true"
```

例 11.35. standalone.conf.bat

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.security.manager -
Djava.security.policy==\path\to\server.policy -
Djboss.home.dir=%JBoss_HOME% -Djboss.modules.policy-
permissions=true"
```

3. 启动域或服务器。

正常地启动域或服务器。

[提交 bug 报告](#)

11.11.3. 关于 Java 安全管理者策略

安全策略

以不同代码类别表示的一系列权限。Java 安全管理者将应用程序的动作请求和安全策略

进行比较。如果某个动作是策略所允许的，那么安全管理者将允许这个动作执行。如果这个动作是策略所不允许的，那么将拒绝这个动作的执行。安全策略可以根据代码的位置、签名或主题的主体来定义权限。

Java 安全管理者和所用的安全策略是用虚拟机选项 `java.security.manager` 和 `java.security.policy` 配置的。

基本信息

安全策略的条目包含了下列配置元素，它们连接至 `policytool`：

CodeBase

产生代码的 URL 位置（不包括主机和域信息）。这个参数是可选的。

SignedBy

密钥库里用来引用签名者（其私有密钥用于为代码签名）的别名。这可以是单个的值，也可以是用逗号隔开的值的列表。如果忽略它，签名的缺席与否都不会影响 Java 安全管理者。

Principals

principal_type/principal_name 对的列表，它必须出现在执行线程的 `principal` 集里。

Principal 条目是可选的。如果忽略它，则表示执行线程的主体不会影响 Java 安全管理者。

Permissions

赋予代码的权限。许多权限是作为 Java EE 规格的一部分提供的。本文档只涵盖由 JBoss EAP 6 提供的其他权限。

提交 bug 报告

11.11.4. 编写 Java 安全性管理者策略

介绍

多数 JDK 和 JRE 版本里都包含一个 `policytool` 程序，它用于创建和编辑 Java 安全管理者安全策略。关于 `policytool` 的详细信息请访问 <http://docs.oracle.com/javase/6/docs/technotes/tools/>。

过程 11.37. 设置新的 Java 安全性管理者策略

1. 启动 `policytool`。

以下列方式之一启动 `policytool` 工具。

o Red Hat Enterprise Linux

在 GUI 或命令行提示下，运行 `/usr/bin/policytool`。

o Microsoft Windows Server

从开始菜单或 Java 安装的 `bin\` 里运行 `policytool.exe`。在不同系统里，其位置可能会不一样。

2. 创建一个策略。

要创建一个策略，请选择 **Add Policy Entry**。添加你需要的参数，然后点击 **Done**。

3. 编辑现有的策略

从现有的策略列表里选择策略，并选择 **Edit Policy Entry** 按钮。然后根据需要编辑相关参数。

4. 删除现有的策略。

从现有的策略列表里选择策略，并选择 **Remove Policy Entry** 按钮。

[提交 bug 报告](#)

11.11.5. 调试安全管理者策略

你可以启用调试信息来帮助你解决和安全策略相关的问题。`java.security.debug` 选项配置和安全相关的信息的级别。`java -Djava.security.debug=help` 命令将产生具有完整调试选项的帮助信息。当解决和安全相关的故障而完全不知道原因时，设置调试级别为 **all** 是很有用的。但对于普通的用途而言，这会产生过多的信息。一般默认的选项是 **access:failure**。

过程 11.38. 启用普通调试

- 这个过程将启用普通级别的和安全相关的调试信息。
在服务器配置文件里添加下列行。
 - 如果 JBoss EAP 6 实例运行在受管域里，这一行将添加到 `bin/domain.conf` () 或 `bin\domain.conf.bat` (Windows) 。
 - 如果 JBoss EAP 6 实例作为独立服务器运行，这一行将添加到 `bin/standalone.conf` () 或 `bin\standalone.conf.bat` (Windows) 。

Linux

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.debug=access:failure"
```

Windows

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.security.debug=access:failure"
```

结果

启用了普通级别的和安全相关的调试信息。

[提交 bug 报告](#)

11.12. SSL 加密

11.12.1. 对 JBoss EAP 6 Web 服务器实施 SSL 加密

介绍

许多 web 应用程序都要求对客户端和服务器间的连接进行 SSL 加密，这也被称为 **HTTPS** 连接。你可以通过这个过程对服务器或服务器组启用 **HTTPS**。

前提条件

- 您需要一系列 SLL 加密密钥和加密证书。你可以从证书签名机构购买或者使用命令行工具来生成。关于使用红帽企业版 Linux 里的可用工具来生成加密密钥，请参考 [第 11.12.2 节“生成 SSL 密钥和证书”](#)。
- 您的环境和设置的细节：

- 证书文件所在的完整目录名。
- 你的加密密钥的密码。
- 连接域控制器或独立服务器的管理 CLI 命令。



注意

这个过程使用适合用于受管域的 JBoss EAP 6 配置的命令。如果你使用的是独立服务器，请从任何管理 CLI 命令的开头将 `/profile=default` 删除。

过程 11.39. 配置 JBoss Web 服务器以使用 HTTPS

1. 添加新的 HTTPS 连接器。

执行下列管理 CLI 命令来修改配置。这会创建一个新的加密连接器，名为 **HTTPS**。它使用 **https** 模式、**https** 套接字绑定（默认为 **8443** 端口），且被设置为安全的。

例 11.36. 管理 CLI 命令

```
/profile=default/subsystem=web/connector=HTTPS/:add(socket-binding=https,scheme=https,protocol=HTTP/1.1,secure=true)
```

2. 执行下列管理 CLI 命令来设置协议为 TLSv1。

例 11.37. 管理 CLI 命令

```
/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=protocol,value=TLSv1)
```

3. 选择合适的加密套件

加密套件由使用大量的加密类型的构建模块组成。第一个表列出了我们推荐的加密类型。第二个表列出可能用来和现有软件兼容的加密类型，它们不具备和我们所推荐的相等的安全性。



警告

Red Hat 推荐在**加密套件 (cipher-suite)** 里使用其白名单上的强密码。启用弱加密会导致重大的安全风险。在确定加密套件至前，请阅读 **JDK** 供应商的文档，因为它们可能会有兼容性问题。

表 11.9. 推荐的加密类型

具有 2048 位密钥和 OAEP 的 RSA
CBC 模式的 AES-128

SHA-256
HMAC-SHA-256
HMAC-SHA-1

表 11.10. 其他加密类型

具有大于 1024 密钥及传统 padding 的 RSA
AES-192
AES-256
3DES（三重 DES，带有两个或三个 56 位的密钥）
RC4（极不推荐）
SHA-1
HMAC-MD5

关于连接器的 SSL 属性参数的完整列表，请参考 [第 11.12.3 节“SSL 连接器引用”](#)。

4. 配置 SSL 加密证书和密钥。

执行下列 CLI 命令来配置你的 SSL 证书，请用自己的值来替换例子中的值。这个例子假设密钥库被复制到服务器的配置目录，对于受管域来说，也就是 `EAP_HOME/domain/configuration/`。

例 11.38. 管理 CLI 命令

```
/profile=default/subsystem=web/connector=HTTPS/ssl=configuration:ad
dd(name=https,certificate-key-
file="${jboss.server.config.dir}/keystore.jks",password=SECRET,
key-alias=KEY_ALIAS, cipher-suite=CIPHERS)
```

5. 部署应用程序。

部署一个使用你已经配置好的配置集的应用程序到服务器组。如果你使用的是独立服务器，将这个应用程序部署至服务器。它的 HTTPS 请求将使用新的 SSL 加密的连接。

[提交 bug 报告](#)

11.12.2. 生成 SSL 密钥和证书

要使用 SSL 加密的 HTTP 协议（HTTPS）以及其他类型的通讯，你需要一个签名的加密证书。你可以从证书认证机构（CA）购买或者使用自签名（self-signed）的证书。许多第三方机构认为自签名的证书是不可靠的，但它适合于内部的测试目的。

这个过程让你使用红帽企业版 Linux 里的工具创建自签名的证书。

必须具备的条件

- 你需要 **keytool** 工具，任何 JDK 都提供它。红帽企业版 Linux 上的 OpenJDK 将这个命令安装在 `/usr/bin/keytool`。
- 请理解 **keytool** 命令的语法和参数。这个过程将使用非常浅显的说明，因为对 SSL 证书或 **keytool** 命令的深入讨论都超出了本文档的范畴。

过程 11.40. 生成 SSL 密钥和证书

1. 用公共和私有密钥生成密钥库。
运行下列命令来当前目录里生成带有别名 **jboss** 的名为 **server.keystore** 的密钥库。

```
keytool -genkeypair -alias jboss -keyalg RSA -keystore
server.keystore -storepass mykeystorepass --dname
"CN=jsmith,OU=Engineering,O=mycompany.com,L=Raleigh,S=NC,C=US"
```

下表描述了用于 **keytool** 命令的参数：

参数	描述
-genkeypair	keytool 命令生成包含公共和私有密钥的密钥对。
-alias	密钥库的别名。这个值是任意的，但别名 jboss 是 JBoss Web 服务器使用的默认值。
-keyalg	密钥对的生成算法。这个例子里是 RSA 。
-keystore	密钥库文件的名称和位置。默认的位置是当前的目录。你可以选择任意名字。在这个例子里是 server.keystore 。
-storepass	这个密码用于针对密钥库进行验证，从而读取密钥。这个密码长度必须至少为 6 且在访问时密钥库时提供。在这个例子里，我们使用 mykeystorepass 。如果你忽略这个参数，在执行命令时你将被提示输入它。
-keypass	这是实际密钥的密码。 <div> 注意 由于实现的限制，它必须和库的密码相同。</div>

参数	描述
--dname	<p>引号括起的描述密钥的可区分名称的字符串，如"CN=jsmith,OU=Engineering,O=mycompany.com,L=Raleigh,C=US"。这个字符串是下列组件的组合：</p> <ul style="list-style-type: none"> ◦ CN - 常用名或主机名。如果主机名是"jsmith.mycompany.com"，那么 CN 就是"jsmith"。 ◦ OU - 机构单元，如 "Engineering"。 ◦ O - 机构名称，如 "mycompany.com"。 ◦ L - 地区，如 "Raleigh" 或 "London"。 ◦ S - 州或省，如 "NC"。这个参数是可选的。 ◦ C - 两个字符的国家代码，如 "US" 或 "UK"。

当你执行上述命令时，你会被提示输入下列信息：

- 如果你在命令行没有使用 **-storepass** 参数，你会被要求输入密钥库的密码。在下次提示时再次输入新密码。
- 如果你在命令行没有使用 **-keypass** 参数，你会被要求输入密钥密码。按 **Enter** 将其设置为和密钥密码相同的值。

当命令执行完毕时，**server.keystore** 文件包含了带有别名 **jboss** 的单个密钥。

2. 检验这个密钥。

通过下列命令检验这个密钥是否正常工作。

```
keytool -list -keystore server.keystore
```

你会被提示输入密钥库密码。密钥库的内容将被显示（这个例子里是名为 **jboss** 的单个密钥）。请注意 **jboss** 密钥的类型是 **keyEntry**。这表示密钥库包含这个密钥的公共和私有条目。

3. 生成一个证书签名请求。

运行下列命令使用步骤一里创建的密钥库里的公共密钥来生成一个证书签名请求。

```
keytool -certreq -keyalg RSA -alias jboss -keystore server.keystore
-file certreq.csr
```

系统会提示你输入密码以针对密钥库进行验证。然后 **keytool** 命令将在当前工作目录里创建一个名为 **certreq.csr** 的证书签名请求。

4. 测试新生成的证书签名请求。

通过下列命令测试证书的内容。

```
openssl req -in certreq.csr -noout -text
```

证书细节将被显示。

5. 可选：提交证书签名请求到证书认证机构（CA）。

证书机构（Certificate Authority，CA）可以验证你的证书，它被第三方的客户认为是可靠的。CA 向你提供一个签名的证书，还有一个或多个可选的中间证书。

6. 可选：从密钥库导出自签名的证书

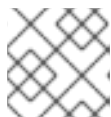
如果你只需要用来进行测试或内部使用，你可以使用自签名的证书。你可以从密钥库导出在步骤一创建的证书：

```
keytool -export -alias jboss -keystore server.keystore -file  
server.crt
```

系统会提示你输入密码以针对密钥库进行验证。名为 **server.crt** 的自签名的证书将在当前工作目录里创建。

7. 导入已签名的证书以及任何中间的证书。

按照 CA 里说明的顺序导入每个证书。对于每个要导入的证书，请用实际的文件名替换 **intermediate.ca** 或 **server.crt**。如果你的证书未作为独立的文件提供，请为每个证书创建一个独立的文件，并将其内容粘贴到文件里。



注意

你的已签名的证书和密钥都是有价值的资产。请小心地在服务器间传递。

```
keytool -import -keystore server.keystore -alias intermediateCA -  
file intermediate.ca
```

```
keytool -importcert -alias jboss -keystore server.keystore -file  
server.crt
```

8. 测试你的证书是否已经成功导入。

运行下列命令，遇提示时输入密钥库密码。密钥库的内容将被显示，证书将是列表里的一部分。

```
keytool -list -keystore server.keystore
```

结果

你签名的证书现在已包含在密钥库里了，它可用来加密 SSL 连接，包括 HTTPS web 服务器通讯。

[提交 bug 报告](#)

11.12.3. SSL 连接器引用

JBoss Web 连接器可能包括了下列 SSL 配置属性。提供的 CLI 命令是针对使用 **default** 配置集的受管域的。按照你的需要修改这个配置集名称（对于受管域），或者忽略命令行的 **/profile=default** 部分（对于独立服务器）。

表 11.11. SSL 连接器属性

属性	描述	CLI 命令
name	SSL 连接器的显示名称。	name 属性是只读的。
verify-client	<p>verify-client 具有不同的值，这取决于是否使用 HTTP/HTTPS 连接器或者是否使用 APR 连接器。</p> <p>HTTP/HTTPS 连接器</p> <p>可能的值有设置为 true、false 或 want。true 表示在接受连接前需要客户的有效证书链。如果你希望 SSL 栈来请求客户证书，可以将其设为 want。设置为 false（默认值）表示不要求证书链，除非客户请求被使用 CLIENT-CERT 验证的安全约束保护的资源。</p> <p>Native APR 连接器</p> <p>可能的值有 optional、require、optionalNoCA 和 none（或与 none 有相同效果的其他字符串）。这些值确定了证书是否是可选的、必需的、不需要证书机构还是完全不需要。默认值是 none，表示客户没有机会提交证书。</p>	<p>第一个命令示例使用了 HTTPS 连接器。</p> <pre>/profile=default/sub system=web/connector =HTTPS/ssl=configuration/:write- attribute(name=verify- client,value=want)</pre> <p>第二个命令示例使用了 APR 连接器。</p> <pre>/profile=default/sub system=web/connector =APR/ssl=configuration/:write- attribute(name=verify- client,value=require)</pre>
verify-depth	中间证书发行者在决定客户是否具有有效证书前检查的最次数。默认值是 10 。	<pre>/profile=default/sub system=web/connector =HTTPS/ssl=configuration/:write- attribute(name=verify- depth,value=10)</pre>
certificate-key-file	保存服务器证书的密钥库文件的完整文件路径和名称。对于 JSSE 加密，这个证书文件将是唯一的，而 OpenSSL 则使用几个文件。默认值是运行 JBoss EAP 6 的用户的主目录下的 .keystore 。如果你的 keystoreType 没有使用文件，请将这个参数设置为空字符串。	<pre>/profile=default/sub system=web/connector =HTTPS/ssl=configuration/:write- attribute(name=certificate- key- file,value=../domain /configuration/serve r.keystore)</pre>

属性	描述	CLI 命令
certificate-file	如果你使用 OpenSSL 加密，请设置这个参数的值为包含服务器证书的文件的完整路径。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=certificate-file,value=server.crt)</pre>
password	用于信任库和密钥库的密码。在下面的例子里，请用自己的密码替换 PASSWORD 。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=password,value=PASSWORD)</pre>
protocol	要使用的 SSL 协议的版本。支持的取值取决于底层的 SSL 实现（JSSE 或 OpenSSL）。请参考 Java SSE 文档 。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=protocol,value=ALL)</pre>

属性	描述	CLI 命令
cipher-suite	<p>允许的加密密码列表。对于 JSSE 语法，它必须是一个用逗号隔开的列表。对于 OpenSSL 语法，它必须是一个用分号隔开的列表。</p> <p>默认值是：HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5。</p> <p>这个例子只列出了两种可能的密码，但实际的例子可能使用更多密码。</p> <div>  <div> <p>重要</p> <p>使用弱密码有重大的安全风险。对于 NIST 推荐的密码套件，请参考 http://www.nist.gov/manuscript-publication-search.cfm?pub_id=915295。</p> </div> </div> <p>对于可用的 OpenSSL 密码列表，请参考 https://www.openssl.org/docs/apps/ciphers.html#CIPHER_STRINGS。请注意下列密码是不被支持的：@SECLEVEL、SUITEB128、SUITEB128ONLY、SUITEB192。</p>	<pre>/profile=default/sub system=web/connector =HTTPS/ssl=configura tion/:write- attribute(name=ciphe r-suite, value="TLS_RSA_WITH_ AES_128_CBC_SHA,TLS_ RSA_WITH_AES_256_CBC _SHA")</pre>
key-alias	<p>用于密钥库里的服务器证书的别名。在下面的例子中，请用你的证书别名替换 KEY_ALIAS。</p>	<pre>/profile=default/sub system=web/connector =HTTPS/ssl=configura tion/:write- attribute(name=key- alias,value=KEY_ALIA S)</pre>
truststore-type	<p>信任库的类型。不同的密钥库包括 PKCS12 和 Java 的标准 JKS。</p>	<pre>/profile=default/sub system=web/connector =HTTPS/ssl=configura tion/:write- attribute(name=trust store- type,value=jks)</pre>

属性	描述	CLI 命令
keystore-type	密钥库的类型那个。不同的密钥库类型包括 PKCS12 和 Java 的标准 JKS 。	<pre>/profile=default/sub system=web/connector =HTTPS/ssl=configura tion/:write- attribute(name=keyst ore-type,value=jks)</pre>
ca-certificate-file	包含 CA 证书的文件。对于 JSSE 是 truststoreFile ，并对密钥库使用相同密码。 ca-certificate-file 文件被用来检验客户证书。	<pre>/profile=default/sub system=web/connector =HTTPS/ssl=configura tion/:write- attribute(name=certi ficate- file,value=ca.crt)</pre>
ca-certificate-password	用于 ca-certificate-file 的证书密码。在下面的例子里，请用自己的掩码密码替换其中的 MASKED_PASSWORD 。	<pre>/profile=default/sub system=web/connector =HTTPS/ssl=configura tion/:write- attribute(name=ca- certificate- password,value=MASKE D_PASSWORD)</pre>
ca-revocation-url	包含撤销列表的文件或 URL。它指向 crlFile (JSSE) 或 SSLCARevocationFile (SSL)。	<pre>/profile=default/sub system=web/connector =HTTPS/ssl=configura tion/:write- attribute(name=ca- revocation- url,value=ca.crl)</pre>
session-cache-size	SSL 会话缓存的大小。这个属性仅用于 JSSE 连接器。默认值是 0 ，它表示缓存大小是无限的。	<pre>/profile=default/sub system=web/connector =HTTPS/ssl=configura tion/:write- attribute(name=sessi on-cache- size,value=100)</pre>

属性	描述	CLI 命令
session-timeout	在缓存的 SSLSession 过期前的秒数。这个属性仅适用于 JSSE 连接器。默认值为 86400 秒，也就是 24 小时。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=session-timeout,value=43200)</pre>

[提交 bug 报告](#)

11.13. 用于敏感字符串的密码库

11.13.1. 关于保护明码文件里的敏感字符

Web 应用程序和其他部署经常含有明码文件，如 XML 部署描述符，其中包含敏感信息如密码和其他敏感字符串。JBoss EAP 6 包含一个密码库（Password Vault）机制，可以让你加密敏感信息并将其存储在一个加密的密钥库（Keystore）里。这个库机制管理解密用于安全域、安全区或其他验证系统的字符串。这提供了一个额外的安全层。这个机制依赖于被支持的 JKD 实现里包含的一些工具。



警告

我们在 JBoss EAP 6 里使用库安全功能时遇到了问题。在使用 IBM JDK 时，我们发现 Sun/Oracle **keytool** 生成的 **vault.keystore** 不是有效的密钥库。这是由于不同的 Java 供应商里 JCEKS 密钥库实现是不同的。

当 Oracle Java 生成的密钥库用于 IBM Java 上的 JBoss EAP 实例时就会出现这个问题。此时服务器不会启动并抛出如下异常：

```
java.io.IOException:
com.sun.crypto.provider.SealedObjectForKeyProtector
```

目前，唯一的变通办法是在使用 IBM Java 实现的环境里避免用 Oracle **keytool** 来生成密钥库。

[提交 bug 报告](#)

11.13.2. 创建一个 Java 密钥库来存储敏感信息

前提条件

- **keytool** 命令必须可用。它是 JRE 提供的命令。请找到这个文件所在的位置，在红帽企业版 Linux 里，它是 **/usr/bin/keytool**。

过程 11.41. 设置 Java 密钥库

1. 创建一个目录来存储你的密钥库和其他加密的信息。

创建一个目录来存储你的密钥库和其他加密的信息。这个过程剩余部分将假设这个目录是 **EAP_HOME/vault/**。既然这个目录将包含敏感信息，它应该只能被有限的用户访问。至少运行 JBoss EAP 的用户帐号需要读-写权限。

2. 确定 keytool 要使用的参数。

确定下列参数：

alias

别名是 **vault** 或其他存储在密钥库里的数据的唯一标识符。这个过程结尾的命令示例里的别名是 **vault**。别名是区分大小写的。

keyalg

用于加密的算法。这个过程里的例子使用了 **RSA**。请查看你的 JRE 和操纵系统的文档，看那种选择是可用的。

keysize

加密密钥的大小影响了通过 **brute force** 解密的难度。这个过程里的例子使用了 **2048**。关于合适的值的信息，请参考 **keytool** 所附带的文档。

keystore

密钥库是一个保存加密信息以及如何解密的信息的数据库。如果你没有指定密钥库，默认的密钥库是你的主目录下的 **.keystore**。当第一次添加数据到密钥库时，它会被创建。这个过程里的例子使用了 **vault.keystore** 密钥库。

keytool 命令有很多其他选项。更多的细节，请参考你的 JRE 或操纵系统的文档。

3. 确定 keystore 命令会询问的问题的答案。

keystore 需要下列信息来填充密钥库条目：

密钥库密码

当你创建一个密钥库时，你必须设置密码。为了和将来的密钥库一起使用，你需要提供密码。请选择一个你可以记住的强密码。密钥库的安全性取决于密码以及它所在的文件系统和操纵系统的安全性。

密钥密码 (可选)

除了密钥库密码，你可以为每个保存的密钥指定一个密码。每次使用这个密钥时都需要输入密码。通常这个功能不会被使用。

名和姓

这和列表里余下的信息可以有助于唯一标识密钥并将其放入一个其他密钥的层次结构里。它完全可以不是一个名字，但应该是两个单词，而且是唯一的。这个过程里的例子使用了 **Accounting Administrator**。按照目录的术语，它成为了证书的 *通用名称 (common name)*。

机构内部门 (Organizational unit)

这是一个标识谁在使用证书的单词。它可以是应用程序或商业单元。这个过程里的例子使用了**AccountingServices**。通常，组或应用程序使用的所有密钥库都使用相同的机构内部部门。

机构

这通常是一个代表你的机构名称的单词。它通常在机构使用的所有证书里都保持相同。这个例子使用了**MyOrganization**。

城市或自治区

你的城市

州或省

你的州或省，或者相等的地区

国家

两个字母的国家代码

所有这些信息将创建一个密钥库和证书的层次结构，确保它们使用一致而唯一的命名结构。

4. 运行 **keytool** 命令，提供你收集的信息。

例 11.39. keytool 命令的输入和输出的例子。

```
$ keytool -genseckey -alias vault -storetype jceks -keyalg AES -
keysize 128 -storepass vault22 -keypass vault22 -validity 730 -
keystore EAP_HOME/vault/vault.keystore
Enter keystore password: vault22
Re-enter new password:vault22
What is your first and last name?
  [Unknown]: Accounting Administrator
What is the name of your organizational unit?
  [Unknown]: AccountingServices
What is the name of your organization?
  [Unknown]: MyOrganization
What is the name of your City or Locality?
  [Unknown]: Raleigh
What is the name of your State or Province?
  [Unknown]: NC
What is the two-letter country code for this unit?
  [Unknown]: US
Is CN=Accounting Administrator, OU=AccountingServices,
O=MyOrganization, L=Raleigh, ST=NC, C=US correct?
  [no]: yes

Enter key password for <vault>
      (RETURN if same as keystore password):
```

结果

在 **EAP_HOME/vault/** 目录里创建了一个名为 **vault.keystore** 的文件。它保存了一个名为 **vault** 的密钥，这个密钥将用来为 JBoss EAP 6 保存加密字符串，如密码。

[提交 bug 报告](#)

11.13.3. 设置密钥库密码的掩码并初始化密码库

前提条件

- [第 11.13.2 节 “创建一个 Java 密钥库来存储敏感信息”](#)

1. 运行 `vault.sh` 命令。

运行 `EAP_HOME/bin/vault.sh`。输入 `0` 启动新的交互式会话。

2. 输入保存加密文件的目录。

这个目录应该只能被有限的用户访问。至少运行 JBoss EAP 的用户帐号需要读-写权限。如果您遵循 [第 11.13.2 节 “创建一个 Java 密钥库来存储敏感信息”](#) 里的步骤，您的密钥库会位于 `EAP_HOME/vault/` 目录里。



注意

不要忘记目录名后面的斜杠。根据操作系统来使用 `/` 或 `\`。

3. 输入密钥库的路径。

输入密钥库文件的完整路径。这个例子使用了 `EAP_HOME/vault/vault.keystore`。

4. 加密密钥库的密码。

下面的步骤加密了密钥库的密码，所以你可以在配置文件和应用程序里安全地使用它了。

a. 输入密钥库的密码。

遇到提示时，输入密钥库的密码。

b. 输入一个 salt 值。

输入 8 个字符的 salt 值。这个 salt 值和下面的迭代计数用于创建哈希值。

c. 输入迭代计数。

输入迭代计数的值。

d. 记录密码掩码信息。

密码掩码、salt 和迭代计数都输出在标准输出里。将其记录在一个安全的位置。攻击者可能用它们来破解密码。

e. 输入 vault 的别名。

遇到提示时，输入 vault 的别名。如果你按照 [第 11.13.2 节 “创建一个 Java 密钥库来存储敏感信息”](#) 来创建 vault，那么别名应该是 `vault`。

5. 退出交互式控制台。

输入 `2` 来退出交互式控制台。

结果

你的密钥库密码已设置掩码，可用于配置文件和部署了。此外，你的 vault 已配置完全且可以使用了。

[提交 bug 报告](#)

11.13.4. 配置 JBoss EAP 6 来使用密码库

介绍

你可以在配置文件里为密码设置掩码和其他敏感属性之前，你需要让 JBoss EAP 6 知晓存储和破解密码的密码库。请按照下列步骤来启用这个功能。

前提条件

- [第 11.13.2 节 “创建一个 Java 密钥库来存储敏感信息”](#)
- [第 11.13.3 节 “设置密钥库密码的掩码并初始化密码库”](#)

过程 11.42. 设置密码库

1. **确定这个命令的正确的值。**
确定下列参数的值，这个命令将用它们来创建密钥库。关于创建密钥库的信息，请参考下列主题：[第 11.13.2 节 “创建一个 Java 密钥库来存储敏感信息”](#) 和 [第 11.13.3 节 “设置密钥库密码的掩码并初始化密码库”](#)。

参数	描述
KEYSTORE_URL	密钥库文件的 URL 的文件系统路径，它通常类似于 <code>vault.keystore</code> 。
KEYSTORE_PASSWORD	用来访问密钥库的密码。这个值应该被标记。
KEYSTORE_ALIAS	密钥库的名称。
SALT	用来加密和解密密钥库值的 Salt。
ITERATION_COUNT	运行加密算法的次数。
ENC_FILE_DIR	密钥库命令运行的目录路径。通常是包含密码库的目录。
host (managed domain only)	你在配置的主机的名称

2. **使用管理 CLI 来启用密码库。**
根据使用的是受管域还是独立服务器配置，分别运行下列命令。用这个步骤里第一步里的值替换命令里的值。



注意

如果你使用的是 Microsoft Windows 服务器，请用额外的 \ 字符将目录路径里的每个 \ 字符转义。例如，`C:\\data\\vault\\vault.keystore`。这是因为单个 \ 仅表示字符转义。

◦ 受管域

```
/host=YOUR_HOST/core-service=vault:add(vault-options=[("KEYSTORE_URL" => "PATH_TO_KEYSTORE"), ("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"), ("KEYSTORE_ALIAS" => "ALIAS"), ("SALT" =>
```

```
"SALT"), ("ITERATION_COUNT" => "ITERATION_COUNT"), ("ENC_FILE_DIR"
=> "ENC_FILE_DIR"]])
```

o 独立服务器

```
/core-service=vault:add(vault-options=[("KEYSTORE_URL" =>
"PATH_TO_KEYSTORE"), ("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"),
("KEYSTORE_ALIAS" => "ALIAS"), ("SALT" => "SALT"),
("ITERATION_COUNT" => "ITERATION_COUNT"), ("ENC_FILE_DIR" =>
"ENC_FILE_DIR"]])
```

下面是带有假定值的命令示例：

```
/core-service=vault:add(vault-options=[("KEYSTORE_URL" =>
"/home/user/vault/vault.keystore"), ("KEYSTORE_PASSWORD" => "MASK-
3y28rCZ1cKR"), ("KEYSTORE_ALIAS" => "vault"), ("SALT" =>
"12438567"), ("ITERATION_COUNT" => "50"), ("ENC_FILE_DIR" =>
"/home/user/vault/")])
```

结果

JBoss EAP 6 已配置好通过密码库来破解掩码字符串。要添加字符串到库里并在配置里使用它们，请参考下列主题：[第 11.13.6 节“在 Java 密钥库里保存和获取加密的敏感字符串”](#)。

[提交 bug 报告](#)

11.13.5. 配置 JBoss EAP 6 来使用自定义的密码库实现

概述

您可以用自己的 **SecurityVault** 实现来遮盖配置文件里的密码和其他敏感属性。

过程 11.43. 使用自定义的密码库实现

1. 创建一个实现 **SecurityVault** 接口的类。
2. 创建包含之前步骤里的类的模块，然后指定 **org.picketbox** 上的依赖关系，其中的接口是 **SecurityVault**。
3. 通过添加 **vault** 元素和下列属性启用 JBoss EAP 服务器配置里的自定义密码库：

code

实现 **SecurityVault** 的类的全限定名。

module

包含自定义类的模块的名称。

或者，您也可以使用 **vault-options** 参数来初始化密码库的自定义类。例如：

```
/core-
service=vault:add(code="custom.vault.implementation.CustomSecurityVa
ult", module="custom.vault.module", vault-options=[("KEYSTORE_URL"
```

```
=> "PATH_TO_KEYSTORE"), ("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"),
("KEYSTORE_ALIAS" => "ALIAS"), ("SALT" => "SALT"), ("ITERATION_COUNT"
=> "ITERATION_COUNT"), ("ENC_FILE_DIR" => "ENC_FILE_DIR"]])
```

结果

配置 JBoss EAP 6 用自定义的密码库实现来解密遮掩的字符串。

[提交 bug 报告](#)

11.13.6. 在 Java 密钥库里保存和获取加密的敏感字符串

概述

在普通文本配置文件里包含密码和其他敏感字符串是不安全的。JBoss EAP 6 包括了在加密的密钥库里存储并为这些敏感字符串设置掩码，以及在配置文件里使用掩码值。

前提条件

- [第 11.13.2 节 “创建一个 Java 密钥库来存储敏感信息”](#)
- [第 11.13.3 节 “设置密钥库密码的掩码并初始化密码库”](#)
- [第 11.13.4 节 “配置 JBoss EAP 6 来使用密码库”](#)
- `EAP_HOME/bin/vault.sh` 应用程序必须可以通过命令行界面来访问。

过程 11.44. 设置 Java 密钥库

1. 运行 `vault.sh` 命令。
运行 `EAP_HOME/bin/vault.sh`。输入 `0` 启动新的交互式会话。
2. 输入保存加密文件的目录。
如果你遵循 [第 11.13.2 节 “创建一个 Java 密钥库来存储敏感信息”](#)，你的密钥库将位于主目录里的 `vault/` 下。在多数情况下，将所有加密信息保存在相同的位置是有意义的。这个例子使用了 `/home/USER/vault/`。



注意

不要忘记目录名后面的斜杠。根据操作系统来使用 `/` 或 `\`。

3. 输入密钥库的路径。
输入密钥库文件的完整路径。这个例子使用了 `/home/USER/vault/vault.keystore`。
4. 输入密钥库密码、Vault 名、Sale 和迭代计数。
遇到提示时，输入密钥库密码、Vault 名、Sale 和迭代计数。然后将执行握手操作。
5. 选择保存密码的选项。
选择选项 `0` 来保存密码或其他敏感字符串。
6. 输入这个值。
遇提示时，将这个值输入两次。如果两次的值不匹配，会提示你再次输入。
7. 输入 Vault Block。
输入 Vault Block，

8. 输入属性名称。

输入你在保存的属性的名称。一个例子就是 **password**。

结果

类似于下面的信息表示这个属性已经被保存了。

安全的属性值已被存储在库里。

9. 记录关于加密字符串的信息。

标准输出的信息将显示 **vault block**、属性名称、共享密钥和在配置里使用字符串的建议。请在安全的地方记录这些信息。下面是输出示例。

```
*****
Vault Block:ds_ExampleDS
Attribute Name:password
Configuration should be done as follows:
VAULT::ds_ExampleDS::password::1
*****
```

10. 在你的配置里使用加密的字符串。

在你的配置里使用前面步骤里的字符串来替代普通文本字符串。下面是使用加密密码的数据源。

```
...
<subsystem xmlns="urn:jboss:domain:datasources:1.0">
  <datasources>
    <datasource jndi-name="java:jboss/datasources/ExampleDS"
enabled="true" use-java-context="true" pool-name="H2DS">
      <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-
1</connection-url>
      <driver>h2</driver>
      <pool></pool>
      <security>
        <user-name>sa</user-name>
        <password>${VAULT::ds_ExampleDS::password::1}</password>
      </security>
    </datasource>
  </drivers>
  <driver name="h2" module="com.h2database.h2">
    <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-
datasource-class>
  </driver>
</drivers>
</datasources>
</subsystem>
...
```

你可以在允许表达式的任何域或独立配置文件里使用加密字符串。



注意

要检查某个子系统里是否允许表达式，请在这个子系统里运行下列 CLI 命令：

```
/host=master/core-service=management/security-  
realm=TestRealm:read-resource-description(recursive=true)
```

在这个命令的输出里，查找 **expressions-allowed** 参数的值。如果它为 **true**，你可以在这个特定子系统的配置里使用表达式。

在密钥库里保存了字符串后，请使用下列语法用加密字符串替代任何明文字符串。

```
${VAULT::VAULT_BLOCK::ATTRIBUTE_NAME::ENCRYPTED_VALUE}
```

下面是一个例子，其 Vault Block 是 **ds_ExampleDS**，而属性是 **password**。

```
<password>${VAULT::ds_ExampleDS::password::1}</password>
```

[提交 bug 报告](#)

11.13.7. 存储和解析应用程序里的敏感字符串

介绍

JBoss EAP 6 的 **Configuration** 元素支持使用安全库（Security Vault）机制根据 Java 密钥库里保存的值来解析加密的字符串。你可以在自己的应用程序里添加对这个功能的支持。

首先，添加这个密码到库里。然后，用保存在库的密码替换明文密码。你可以使用这个方法隐藏应用程序里的任何敏感字符串。

前提条件

在执行这个过程之前，请确保保存库文件的目录是存在的。将其保存在什么位置是无所谓的，只要执行 JBoss EAP 6 的用户有读写这些文件的权限。这个例子将 **vault/** 目录放到 **/home/USER/vault/** 目录里。这个库本身也是 **vault/** 里一个名为 **vault.keystore** 的文件。

例 11.40. 添加密码字符串到库里

请用 **EAP_HOME/bin/vault.sh** 命令添加这个字符串到库里。下面的屏幕输出包含了这个命令的完整过程和结果。用户输入的值被高亮显示。出于格式的考虑，我们删除了一些输出。在 **Microsoft Windows** 里，这个命令是 **vault.bat**。请注意，在 **Microsoft Windows** 里，文件路径要使用 **** 来分隔，而不是 **/**。

```
[user@host bin]$ ./vault.sh
*****
****   JBoss Vault   ****
*****

Please enter a Digit::    0: Start Interactive Session  1: Remove
Interactive Session  2: Exit
0
Starting an interactive session
Enter directory to store encrypted files:/home/user/vault/
Enter Keystore URL:/home/user/vault/vault.keystore
Enter Keystore password: ...
```



```

Enter Keystore password again: ...
Values match
Enter 8 character salt:12345678
Enter iteration count as a number (Eg: 44):25

Enter Keystore Alias:vault
Vault is initialized and ready for use
Handshake with Vault complete
Please enter a Digit:: 0: Store a password 1: Check whether password
exists 2: Exit
0
Task: Store a password
Please enter attribute value: sa
Please enter attribute value again: sa
Values match
Enter Vault Block:DS
Enter Attribute Name:thePass
Secured attribute value has been stored in vault.

Please make note of the following:
*****
Vault Block:DS
Attribute Name:thePass
Configuration should be done as follows:
VAULT::DS::thePass::1
*****

Please enter a Digit:: 0: Store a password 1: Check whether password
exists 2: Exit
2

```

将被加入到 Java 代码里的字符串是输出结果里最后面的值，就是以 **VAULT** 开始的行。

下列 **servlet** 使用了 **vaulted** 字符串而不是明文密码。明文版本被注释了，你可以看到其中的区别。

例 11.41. 使用 vaulted 密码的 servlet

```

package vaulterror.web;

import java.io.IOException;
import java.io.Writer;

import javax.annotation.Resource;
import javax.annotation.sql.DataSourceDefinition;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;

/*@DataSourceDefinition(
    name = "java:jboss/datasources/LoginDS",
    user = "sa",

```

```

        password = "sa",
        className = "org.h2.jdbcx.JdbcDataSource",
        url = "jdbc:h2:tcp://localhost/mem:test"
    )*/
    @DataSourceDefinition(
        name = "java:jboss/datasources/LoginDS",
        user = "sa",
        password = "VAULT::DS::thePass::1",
        className = "org.h2.jdbcx.JdbcDataSource",
        url = "jdbc:h2:tcp://localhost/mem:test"
    )
    @WebServlet(name = "MyTestServlet", urlPatterns = { "/my/" },
        loadOnStartup = 1)
    public class MyTestServlet extends HttpServlet {

        private static final long serialVersionUID = 1L;

        @Resource(lookup = "java:jboss/datasources/LoginDS")
        private DataSource ds;

        @Override
        protected void doGet(HttpServletRequest req, HttpServletResponse
            resp) throws ServletException, IOException {
            Writer writer = resp.getWriter();
            writer.write((ds != null) + "");
        }
    }

```

你的 `servlet` 现在可以解析 `vaulted` 字符串了。

[提交 bug 报告](#)

11.14. FIPS 140-2 兼容加密

11.14.1. 关于 FIPS 140-2 兼容性

联邦信息处理标准 140-2 (Federal Information Processing Standard, FIPS 140-2) 是鉴定加密软件模块的美国政府计算机安全标准。FIPS 140-2 兼容性通常是政府机构和私有企业使用的软件系统的要求之一。

JBoss EAP 6 使用了外部的模块加密，你可以配置它使用兼容 FIPS 140-2 的加密模块。

[提交 bug 报告](#)

11.14.2. 兼容 FIPS 140-2 的密码

兼容 FIPS 的密码必须具有下列特征：

1. 其长度必须至少有 7 个字符。
2. 必须包含来自至少三个下列字符类别里的字符：
 - ASCII 数字

- 小写的 ASCII 字符
- 大写的 ASCII 字符
- 非字母的 ASCII，和
- 非 ASCII 字符

如果第一个字符是大写的 ASCII 字符，它不会算作限制 2 里的大写 ASCII 字符。

如果密码的最后一个字符是 ASCII 数字，它不会算作限制 2 里的 ASCII 数字。

[提交 bug 报告](#)

11.14.3. 在红帽企业版 Linux 6 上启用 SSL 的 FIPS 140-2 加密

这个任务描述了如何配置 JBoss EAP 6 的 Web 容器（JBoss Web）的 SSL 为兼容 FIPS 140-2 的加密。这个任务只涵盖在红帽企业版 Linux 6 上实现的步骤。

为了这个功能，这个任务使用了 FIPS 模式的 Mozilla NSS 库。

前提条件

- 红帽企业版 Linux 6 必须已经配置为 FIPS 140-2 兼容的模式。请参考 <https://access.redhat.com/knowledge/solutions/137833>。

过程 11.45. 启用 SSL 的 FIPS 140-2 兼容加密

1. 创建数据库

在 **jboss** 用户拥有的一个目录里创建 NSS 数据库。

```
$ mkdir -p /usr/share/jboss-as/nssdb
$ chown jboss /usr/share/jboss-as/nssdb
$ modutil -create -dbdir /usr/share/jboss-as/nssdb
```

2. 创建 NSS 配置文件

在 **/usr/share/jboss-as** 目录里创建一个名为 **nss_pkcs11_fips.cfg** 的文本文件，它具有下列内容：

```
name = nss-fips
nssLibraryDirectory=/usr/lib64
nssSecmodDirectory=/usr/share/jboss-as/nssdb
nssModule = fips
```

NSS 配置文件必须指定：

- 名称
- NSS 库所在的目录，和
- 步骤 1 里创建 NSS 数据库用到的目录。

如果你使用的是红帽企业版 Linux 6 的 64 位版本，请设置 **nssLibraryDirectory** 为 **/usr/lib** 而不是 **/usr/lib64**。

3. 启用 SunPKCS11 供应商

编辑你的 JRE 的 `java.security` 配置文件（`$JAVA_HOME/jre/lib/security/java.security`）并添加下列行：

```
security.provider.1=sun.security.pkcs11.SunPKCS11 /usr/share/jboss-as/nss_pkcs11_fips.cfg
```

请注意这一行里指定的配置文件就是我们在步骤 2 里创建的文件。

这个文件里的任何其他 `security.provider.X` 行都必须将 `X` 递增以设置对应供应商的优先级。

4. 为 NSS 库启用 FIPS 模式

运行 `modutil` 命令以启用 FIPS 模式：

```
modutil -fips true -dbdir /usr/share/jboss-as/nssdb
```

请注意这里指定的目录就是我们在步骤 1 里创建的目录。

此时你可能遇到一个安全库错误，要求你为某些 NSS 共享对象重新生成库签名。

5. 修改 FIPS 令牌的密码

用下列命令在 FIPS 令牌上设置密码。请注意，令牌的名称必须为 **NSS FIPS 140-2 Certificate DB**。

```
modutil -change pw "NSS FIPS 140-2 Certificate DB" -dbdir /usr/share/jboss-as/nssdb
```

用于 FIPS 令牌的密码必须是兼容 FIPS 的密码。

6. 使用 NSS 工具创建证书

输入下列命令来用 NSS 工具创建证书。

```
certutil -S -k rsa -n jbossweb -t "u,u,u" -x -s "CN=localhost, OU=MYOU, O=MYORG, L=MYCITY, ST=MYSTATE, C=MY" -d /usr/share/jboss-as/nssdb
```

7. 配置 HTTPS 连接器使用 PKCS11 密钥库

用 JBoss CLI 工具里的下列命令添加一个 HTTPS 连接器：

```
/subsystem=web/connector=https/:add(socket-binding=https,scheme=https,protocol=HTTP/1.1,secure=true)
```

然后用下列命令添加 SSL 配置，用兼容 FIPS 的密码（参考步骤 5）替换这里的 `PASSWORD`。

```
/subsystem=web/connector=https/ssl=configuration:add(name=https,password=PASSWORD,keystore-type=PKCS11,cipher-suite="SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_DSS_WITH_AES_128_CBC_SHA,TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_DHE_DSS_WITH_AES_256_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,
```

```

    TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDH_ECDSA_WITH_AES_128_CBC
    _SHA,
    TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_ECDSA_WITH_3DES_EDE_CB
    C_SHA,
    TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_256_CB
    C_SHA,
    TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
    ,
    TLS_ECDH_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SH
    A,
    TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_CBC_SH
    A,
    TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA, TLS_ECDH_anon_WITH_AES_128_CBC_S
    HA,
    TLS_ECDH_anon_WITH_AES_256_CBC_SHA")

```

8. 检验

运行下列命令检验 JVM 是否可以从 PKCS11 密钥库楼里读取私有密钥：

```
keytool -list -storetype pkcs11
```

例 11.42. 使用 FIPS 140-2 的 HTTPS 连接器的 XML 配置

```

<connector name="https" protocol="HTTP/1.1" scheme="https" socket-
binding="https" secure="true">
  <ssl name="https" password="*****"
    cipher-
suite="SSL_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
      TLS_RSA_WITH_AES_128_CBC_SHA,
      TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
      TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,

      TLS_DHE_DSS_WITH_AES_256_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA,

      TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
      ,

      TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SH
      A,

      TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SH
      A,

      TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDH_RSA_WITH_AES_128_CBC_SHA,

      TLS_ECDH_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,

      TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,

      TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA, TLS_ECDH_anon_WITH_AES_128_CBC_SHA,
      TLS_ECDH_anon_WITH_AES_256_CBC_SHA"
    keystore-type="PKCS11"/>
</connector>

```

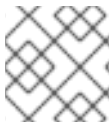
请注意，**cipher-suite** 属性里插入了断行符以便于阅读。

[提交 bug 报告](#)

11.14.4. 在 Apache HTTP 服务器里启用 FIPS 140-2 加密

您可以在 Apache HTTP 服务器配置文件 (`httpd.conf` 或 `ssl.conf`) 里插入 ***SSLFIPS on*** 指令来启用 FIPS 140-2 加密。这个指令必须在 **VirtualHost** 配置部分之外使用。

SSLFIPS on 指令激活了 SSL 库的 `FIPS_mod` 标记。这个模式适用于所有的 SSL 库操作。在添加这个指令后，您需要重启 Apache HTTP 服务器以使其生效。



注意

要启用 FIPS，您必须在系统里安装兼容 FIPS 的 OpenSSL（它支持 ***FIPS_mode*** 标记）。

[提交 bug 报告](#)

第 12 章 安全管理引用

12.1. 包括的验证模块

JBoss EAP 6 里包含了下面的验证模块。其中一些模块处理授权及验证。这些模块的 **Code** 名称通常包含 **Role**。

当你配置这些模块时，请使用 **Code** 值或完整名称（软件包限定）来引用模块。

验证模块

- [表 12.1 “RealmDirect”](#)
- [表 12.2 “RealmDirect 模块选项”](#)
- [表 12.3 “Client”](#)
- [表 12.4 “Client 模块选项”](#)
- [表 12.5 “Remoting”](#)
- [表 12.6 “Remoting 模块选项”](#)
- [表 12.7 “Certificate”](#)
- [表 12.8 “Certificate Module Options”](#)
- [表 12.9 “CertificateRoles”](#)
- [表 12.10 “CertificateRoles 模块选项”](#)
- [表 12.11 “Database”](#)
- [表 12.12 “Database 模块选项”](#)
- [表 12.13 “DatabaseCertificate”](#)
- [表 12.14 “DatabaseCertificate 模块选项”](#)
- [表 12.15 “Identity”](#)
- [表 12.16 “Identity 模块选项”](#)
- [表 12.17 “Ldap”](#)
- [表 12.18 “Ldap 模块选项”](#)
- [表 12.19 “LdapExtended”](#)
- [表 12.20 “LdapExtended 模块选项”](#)
- [表 12.21 “RoleMapping”](#)
- [表 12.22 “RoleMapping 模块选项”](#)

- [表 12.23 “RunAs”](#)
- [表 12.24 “RunAs 选项”](#)
- [表 12.25 “Simple”](#)
- [表 12.26 “ConfiguredIdentity”](#)
- [表 12.27 “ConfiguredIdentity 模块选项”](#)
- [表 12.28 “SecureIdentity”](#)
- [表 12.29 “SecureIdentity 模块选项”](#)
- [表 12.30 “PropertiesUsers”](#)
- [表 12.31 “SimpleUsers”](#)
- [表 12.32 “LdapUsers”](#)
- [表 12.33 “Kerberos”](#)
- [表 12.34 “Kerberos 模块选项”](#)
- [表 12.35 “SPNEGO”](#)
- [表 12.36 “SPNEGO 模块选项”](#)
- [表 12.37 “AdvancedLdap”](#)
- [表 12.38 “AdvancedLdap 模块选项”](#)
- [表 12.39 “AdvancedADLdap”](#)
- [表 12.40 “UsersRoles”](#)
- [表 12.41 “UsersRoles 模块选项”](#)
- [自定义验证模块](#)

表 12.1. RealmDirect

Code	RealmDirect
类	<code>org.jboss.as.security.RealmDirectLoginModule</code>
描述	直接和安全区交互的登录模块实现。这个登录模块允许将所有和后备存储的交互委托给安全区，从而不需要任何重复和同步的定义。它适用于远程调用和管理接口。

表 12.2. RealmDirect 模块选项

选项	类型	默认值	描述
realm	字符串	ApplicationRealm	期望的区的名称。

表 12.3. Client

Code	Client
类	org.jboss.security.ClientLoginModule
描述	这个登录模块的目的是当 JBoss EAP 6 充当客户时建立调用者标识符和凭证。它不应该作为用于服务器验证的安全域的一部分。

表 12.4. Client 模块选项

选项	类型	默认值	描述
multi-threaded	true 或 false	false	如果每个线程都有自己的 principal 和凭证存储，请将其设置为 true 。 false 则指定虚拟机里的所有线程都共享系统的标识符和凭证。
password-stacking	useFirstPass 或 false	false	设置为 useFirstPass 表示这个登录模块应该寻找存储在 LoginContext 里的信息以用作标识符。当堆积这个登录模块和其他模块时可以使用这个选项。
restore-login-identity	true 或 false	false	如果在 login() 方法的开始遇到的标识符和凭证在 logout() 被调用后要重新存储，请将其设置为 true 。

表 12.5. Remoting

Code	Remoting
类	org.jboss.as.security.remoting.RemotingLoginModule
描述	这个登录模块用于检查当前验证的请求是否是从远程连接接收的，如果是则使用在远程验证过程中创建的标识符并和当前请求相关联。如果这个请求不是通过远程连接到达的，这个模块将不会做任何事情，并允许基于 JAAS 的登录继续，进入下一个模块。

表 12.6. Remoting 模块选项

选项	类型	默认值	描述
password-stacking	useFirstPass 或 false	false	useFirstPass 的值表示这个登录模块应该首先查看存储在 LoginContext 里关于这个标识符的信息。当堆积这个登录模块和其他模块时可以使用这个选项。
principalClass	全限定类名。	无	包含一个将 String 参数用作主体 (Principal) 名称的构造器的 Principal 实现类。
unauthenticatedIdentity	主体名称。	无	定义分配给不包含验证信息的请求的主体名称。这允许不受保护的 servlet 调用不要求专有角色的 EJB 上的方法。这样的主体没有关联的角色且只访问未设置安全性的 EJB 或者和 unchecked permission 约束关联的 EJB 。

表 12.7. Certificate

Code	Certificate
类	org.jboss.security.auth.spi.BaseCertLoginModule
描述	这个登录模块的目的是基于 X509 Certificates 验证用户。其中一个用例是 web 应用程序的 CLIENT-CERT 验证。

表 12.8. Certificate Module Options

选项	类型	默认值	描述
securityDomain	字符串	无	具有持有信任证书的信任库的 JSSE 配置的安全域的名称。
verifier	类	无	用户登录证书检验的 org.jboss.security.auth.certs.X509CertificateVerifier 的类名。

表 12.9. CertificateRoles

Code	CertificateRoles
类	org.jboss.security.auth.spi.CertRoleSLoginModule
描述	这个登录模块扩展了 Certificate 登录模块以从属性文件添加角色映射能力。它使用和 Certificate 登录模块相同的所有选项，并添加了如下选项。

表 12.10. CertificateRoles 模块选项

选项	类型	默认值	描述
rolesProperties	字符串	roles.properties	<p>包含分配给每个用户的资源或文件的名称。角色属性文件里的格式必须是 username=role1,role2，其中 username 是证书的 DN，不包括任何 =（等号）和空格字符。下面的例子是正确的格式：</p> <pre>CN\=unit- tests-client,\ OU\=Red\ Hat\ Inc.,\ O\=Red\ Hat\ Inc.,\ ST\=North\ Carolina,\ C\=US</pre>
defaultRolesProperties	字符串	defaultRoles.properties	如果未找到 rolesProperties 文件所使用的资源或文件的名称。
roleGroupSeparator	单个字符。	. (单一句号)	用作 rolesProperties 文件里角色组分隔符的字符。

表 12.11. Database

Code	Database
类	org.jboss.security.auth.spi.DatabaseServerLoginModule

描述	<p>支持验证和角色映射的基于 JDBC 的登录模块。这基于具有下列定义的两个逻辑表。</p> <ul style="list-style-type: none"> • Principals: PrincipalID (text), Password (text) • Roles: PrincipalID (text), Role (text), RoleGroup (text)
----	---

表 12.12. Database 模块选项

选项	类型	默认值	描述
digestCallback	全限定类名	无	包括 pre/post 摘要内容（如对输入密码哈希加密的 salt）的 DigestCallback 实现的类名。它只有在已指定 hashAlgorithm 时才会被使用。
dsJndiName	JNDI 资源	无	保存验证信息的 JNDI 资源的名称。这个选项是必需的。
hashAlgorithm	String	使用明文密码	用于哈希加密密码的消息摘要算法。所支持的算法依赖于 Java 安全提供者，但下列算法都是被支持的： MD5 、 SHA-1 和 SHA-256 。
hashCharset	String	平台的默认编码	转换密码字符串为字节队列所使用的字符集/编码的名称。其值包括所有被支持的 Java 字符集名称。
hashEncoding	String	Base64	所使用的字符串编码格式。
ignorePasswordCase	布尔值	false	指定密码比较是否应该忽略大小写的标记。
inputValidator	全限定类名	无	用于检验客户提供的用户和密码的 InputValidator 实现的实例。
principalsQuery	prepared SQL 语句	select Password from Principals where PrincipalID= ?	获取 principal 的信息的 prepared SQL 查询。
rolesQuery	prepared SQL 语句	无	获取角色信息的 prepared SQL 查询。它应该和 select Role, RoleGroup from Roles where PrincipalID=? 相等，这里的 Role 是角色名称而 RoleGroup 的值总是带有大写 R 的 Roles 或 CallerPrincipal 。

选项	类型	默认值	描述
storeDigestCallback	全限定类名	无	The class name of the DigestCallback implementation that includes pre/post digest content like salts for hashing the store/expected password. Only used if hashStorePassword or hashUserPassword is true and hashAlgorithm has been specified.
suspendResume	布尔值	true	在数据库操作期间现有的 JTA 事务是否应该被暂停。
throwValidatorError	布尔值	false	指定检验错误是否应该开放给客户的标记
transactionManagerJndiName	JNDI 资源	java:/TransactionManager	登录模块使用的事务管理者的 JNDI 名称。

表 12.13. DatabaseCertificate

Code	DatabaseCertificate
类	org.jboss.security.auth.spi.DatabaseCertLoginModule
描述	这个登录模块扩展了 Certificate 登录模块以从数据库表添加角色映射能力。它使用和 Certificate 登录模块相同的所有选项，并添加了如下选项。

表 12.14. DatabaseCertificate 模块选项

选项	类型	默认值	描述
dsJndiName	JNDI 资源		保存验证信息的 JNDI 资源的名称。这个选项是必需的。
rolesQuery	prepared SQL 语句	select Role, RoleGroup from Roles where PrincipalID=?	为了映射角色的 prepared SQL 语句。它应该和 select Role, RoleGroup from Roles where PrincipalID=? 相等，这里的 Role 是角色名称而 RoleGroup 的值总是带有大写 R 的 Roles 或 CallerPrincipal 。

选项	类型	默认值	描述
suspendResume	true 或 false	true	在数据库操作期间现有的 JTA 事务是否应该被暂停。

表 12.15. Identity

Code	Identity
类	org.jboss.security.auth.spi.IdentityLoginModule
描述	关联这个模块选项里指定的 principal 和任何针对这个模块验证的主题。所使用的 Principal 类的类型是 org.jboss.security.SimplePrincipal 。如果没有指定 principal 选项，那使用的名称是 guest 。

表 12.16. Identity 模块选项

选项	类型	默认值	描述
principal	String	guest	用于 principal 的名称。
roles	用逗号隔开的字符串的列表	无	将分配给主题的用逗号隔开的角色列表。

表 12.17. Ldap

Code	Ldap
类	org.jboss.security.auth.spi.LdapLoginModule
描述	当用户名和密码存储在可通过 JNDI LDAP 供应商访问的 LDAP 服务器上时进行的验证。许多选项不是必需的，因为它们可由 LDAP 供应商或系统环境来决定。

表 12.18. Ldap 模块选项

选项	类型	默认值	描述
java.naming.factory.initial	类名	com.sun.jndi.ldap.LdapCtxFactory	InitialContextFactory 实现的类名。
java.naming.provider.url	ldap:// URL	无	LDAP 服务器的 URL。

选项	类型	默认值	描述
<code>java.naming.security.authentication</code>	<code>none</code> 、 <code>simple</code> 或 SASL 机制的名称。	<code>simple</code>	用于绑定 LDAP 服务器的安全级别。
<code>java.naming.security.protocol</code>	传输协议	如果未指定，则由供应商决定。	用于安全访问的传输协议，如 SSL。
<code>java.naming.security.principal</code>	字符串	无	用于验证调用者的 <code>principal</code> 的名称。它是根据下面描述的属性构建的。
<code>java.naming.security.credentials</code>	凭证类型	无	验证模式使用的凭证类型。其中一些例子包括哈希密码、明文密码、密钥或证书。如果没有指定这个属性，其行为将由服务供应商决定。
<code>principalDNPrefix</code>	字符串		添加到用户名以组成用户 DN 的前缀。你可以提示用户输入用户名并使用 <code>principalDNPrefix</code> 和 <code>principalDNSuffix</code> 构建全限定 DN。
<code>principalDNSuffix</code>	字符串		添加到用户名以组成用户 DN 的后缀。你可以提示用户输入用户名并使用 <code>principalDNPrefix</code> 和 <code>principalDNSuffix</code> 构建全限定 DN。
<code>useObjectCredential</code>	<code>true</code> 或 <code>false</code>	<code>false</code>	凭证是否应该用 <code>org.jboss.security.auth.callback.ObjectCallback</code> 类型的回调方法作为不透明的对象、还是用 JAAS <code>PasswordCallback</code> 作为 <code>char[]</code> （字符数组）密码获得。这允许传递 <code>non-char[]</code> （非字符数组）凭证信息到 LDAP 服务器。
<code>rolesCtxDN</code>	全限定的 DN	无	用于搜索用户角色的上下文的全限定 DN。

选项	类型	默认值	描述
userRolesCtxDNAttribute	attribute	无	包含搜索用户角色的上下文 DN 的用户对象里的属性。它和 rolesCtxDN 的区别是搜索用户角色的上下文可能对于每个用户都是唯一的。
roleAttributeID	attribute	roles	包含用户角色的属性的名称。
roleAttributeIsDN	true 或 false	false	roleAttributeID 是否包含角色对象的全限定 DN。如果为 false ，角色名将从上下文名称的 roleNameAttributeID 属性值里获取。某些目录模式，如 Microsoft Active Directory ，要求这个属性的值为 true 。
roleNameAttributeID	attribute	name	包含角色名称的 roleCtxDN 上下文里的属性的名称。如果 roleAttributeIsDN 属性为 true ，这个属性将被用来查找角色对象的 name 属性。
uidAttributeID	attribute	uid	UserRolesAttributeDN 里对应用户 ID 的属性的名称。它被用来定位用户角色。
matchOnUserDN	true 或 false	false	对用户角色搜索是否应该匹配用户的全限定 DN 或只是用户名而已。如果为 true ，完整的用户 DN 将作为匹配值。如果为 false ，则只使用用户名来匹配 uidAttributeName 属性。
allowEmptyPasswords	true 或 false	false	是否允许空的密码。多数 LDAP 服务器将空密码视同匿名登录尝试。要拒绝空密码，请将它设置为 false 。

表 12.19. LdapExtended

Code	LdapExtended
------	---------------------

类	<code>org.jboss.security.auth.spi.LdapExtLoginModule</code>
描述	<p>另外的一个使用搜索来定位绑定用户和关联角色的 LDAP 登录模块实现。角色队列递归地解析 DN 来导航分层的角色结构。它使用和 LDAP 模块相同的 java.naming 选项以及下列 LDAP 模块没有的选项。</p> <p>这个验证以两步进行：</p> <ol style="list-style-type: none"> 1. 对 LDAP 服务器的初始绑定是使用 bindDN 和 bindCredential 选项来完成的。bindDN 是一个具有搜索 baseCtxDN 和 rolesCtxDN 树里的用户和角色能力的 LDAP 用户。要验证的用户 DN 是通过 baseFilter 属性指定的过滤器来进行查询的。 2. 结果用户 DN 是通过将用户 DN 作为 InitialLdapContext 环境的 Context.SECURITY_PRINCIPAL 绑定到 LDAP 服务器来验证的。Context.SECURITY_CREDENTIALS 属性被设置为回调处理程序获得的 String 型的密码。

表 12.20. LdapExtended 模块选项

选项	类型	默认值	描述
baseCtxDN	全限定的 DN	无	开始用户搜索的顶层上下文的固定 DN。
bindCredential	字符串，可以进行加密。	无	更多信息请参考《JBoss EAP 安全指南》
bindDN	全限定的 DN	无	用户和角色查询里用来绑定 LDAP 服务器的 DN。这个 DN 需要读取和搜索 baseCtxDN 和 rolesCtxDN 值上的权限。
baseFilter	LDAP 过滤器字符串	无	用来定位要验证的用户的上下文的搜索过滤器。从登录模块回调方法里获得的输入用户名或 userDN 将替换至过滤器里的 {0} 表达式。搜索过滤器的一个常见例子是 (uid={0}) 。

选项	类型	默认值	描述
rolesCtxDN	全限定的 DN	无	用于搜索用户角色的上下文的固定 DN。这不是实际角色的 DN，它是包含用户角色的对象所在的 DN。例如，在 Microsoft Active Directory 服务器里，它是用户帐号所在的 DN。
roleFilter	LDAP 过滤器字符串	无	用来定位和验证用户相关联的角色的搜索过滤器。从登录模块回调方法里获得的输入用户名和 userDN 将被替换过滤器里的 {0} 表达式。已验证的 userDN 将替换过滤器里的 {1} 表达式。匹配输入用户名的搜索过滤器示例是 (member={0}) 。对应已验证的 userDN 的例子是 (member={1}) 。
roleAttributeIsDN	true 或 false	false	roleAttributeID 是否包含角色对象的全限定 DN。如果为 false ，角色名将从上下文名称的 roleNameAttributeID 属性值里获取。某些目录模式，如 Microsoft Active Directory，要求这个属性的值为 true 。
defaultRole	角色名称	无	用于所有已验证用户的角色
parseRoleNameFromDN	true 或 false	false	指定查询返回的 DN 是否包含 roleNameAttributeID 。如果设置为 true ，将检查 DN 里是否有 roleNameAttributeID ，如果为 false ，将不会检查。这个标记可以提高 LDAP 查询的性能。

选项	类型	默认值	描述
parseUsername	true 或 false	false	指定 DN 是否对用户名进行解析的标记。如果为 true ，DN 将对用户名进行解析。如果为 false ，DN 将不对用户名进行解析。这个选项是和 usernameBeginString 及 usernameEndString 一起使用的。
usernameBeginString	字符串	无	定义将从 DN 的开头删除以显示用户名的字符串。这个选项是和 usernameEndString 一起使用的。
usernameEndString	字符串	无	定义将从 DN 的结尾删除以显示用户名的字符串。这个选项是和 usernameBeginString 一起使用的。
roleNameAttributeID	attribute	name	包含角色名称的 roleCtxDN 上下文里的属性的名称。如果 roleAttributeIsDN 属性为 true ，这个属性将被用来查找角色对象的 name 属性。
distinguishedNameAttribute	attribute	distinguishedName	包含用户 DN 的用户条目里的属性的名称。如果用户自身的 DN 包含特殊字符（如反斜杠）而阻止了正确的用户映射，这就是有必要的。如果这个属性不存在，条目的 DN 将会被使用。
roleRecursion	整数	0	角色搜索的递归级别数。禁用递归可将其设置为 0 。
searchTimeLimit	整数	10000 (10 秒)	用户或角色搜索的超时时间（毫秒）。
searchScope	OBJECT_SCOPE , ONELEVEL_SCOPE , SUBTREE_SCOPE 中的一个	SUBTREE_SCOPE	使用的搜索作用域

选项	类型	默认值	描述
allowEmptyPasswords	true 或 false	false	是否允许空的密码。多数 LDAP 服务器将空密码视同匿名登录尝试。要拒绝空密码，请将它设置为 false 。
referralUserAttributeIDToCheck	attribute	无	如果您没有使用 Referral ，这个选项可被忽略。在使用 Referral 时，如果角色对象位于 Referral 内部，这个选项表示包含为某个角色（例如， member ）定义的用户属性名称。用户将根据这个属性名称的上下文进行检查。如果没有设置这个选项，检查总是会失败，所以角色对象无法存储在 Referral 树里。

表 12.21. RoleMapping

Code	RoleMapping
类	org.jboss.security.auth.spi.RoleMappingLoginModule
描述	映射作为验证过程的最终结果的角色到声明式角色。当你添加这个模块到安全域里时，它必须标记为 optional 。

表 12.22. RoleMapping 模块选项

选项	类型	默认值	描述
rolesProperties	属性文件或资源的全限定文件路径	none	映射角色到替代角色的属性文件或资源的全限定文件路径。其格式是 original_role=role1,role2,role3 。
replaceRole	true 或 false	false	是否添加当前的角色，或者用映射的角色替换当前的角色。设为 true 则进行替换。



注意

对于 **RoleMapping**，**rolesProperties** 模块选项是必需的。

表 12.23. RunAs

Code	RunAs
类	<code>org.jboss.security.auth.spi.RunAsLoginModule</code>
描述	这是一个 Helper 模块，它在验证的登录阶段将 run as 角色推入栈，并在提交或中止阶段从栈里弹出 run as 角色。这个登录模块为其他必须访问安全资源以执行验证的登录模块（如访问安全 EJB 的登录模块）提供了一个角色。在要求 run as 角色的登录模块建立之前，你必须先配置好 RunAsLoginModule 。

表 12.24. RunAs 选项

选项	类型	默认值	描述
roleName	角色名称	nobody	在登录阶段用作 run as 角色的角色的名称。
principalName	主体名称	nobody	登录阶段用作 run as 主体的主体名称。如果没有指定，则使用默认值 nobody 。
principalClass	全限定类名。	无	包含一个将 String 参数用作主体 (Principal) 名称的构造器的 Principal 实现类。

表 12.25. Simple

Code	Simple
类	<code>org.jboss.security.auth.spi.SimpleServerLoginModule</code>
描述	<p>用于测试目的的快速设置安全性的模块。它实现了下列简单的算法：</p> <ul style="list-style-type: none"> 如果密码为 null，验证用户并分配一个 guest 标识符和 guest 角色。 否则，如果密码和用户相同，分配一个和用户名相同的标识符以及 admin 和 guest 角色。 否则，验证将会失败。

Simple 模块选项

Simple 模块没有选项。

表 12.26. ConfiguredIdentity

Code	ConfiguredIdentity
类	org.picketbox.datasource.security.ConfiguredIdentityLoginModule
描述	关联这个模块选项里指定的 principal 和任何针对这个模块验证的主题。所使用的 Principal 类的类型是 org.jboss.security.SimplePrincipal。

表 12.27. ConfiguredIdentity 模块选项

选项	类型	默认值	描述
username	字符串	无	用于验证的用户名。
password	加密的字符串	""	用于验证的密码。要加密这个密码，请在命令行直接使用这个模块。 <div><pre>java org.picketbox. datasource.sec urity.SecureId entityLoginMod ule password_to_en crypt</pre></div> 将这个命令的运行结果粘贴到模块选项的 value 字段。默认值是一个空字符串。
principal	主体的名称	none	将和针对这个模块验证的任何主题关联的 principal。

表 12.28. SecureIdentity

Code	SecureIdentity
类	org.picketbox.datasource.security.SecureIdentityLoginModule
描述	提供这个模块只是为了和之前的系统兼容。它允许你加密密码并和静态 principal 一起使用这个密码。如果你的应用程序使用了 SecureIdentity，请考虑使用密码库机制。

表 12.29. SecureIdentity 模块选项

选项	类型	默认值	描述
username	字符串	无	用于验证的用户名。
password	加密的字符串	""	<p>用于验证的密码。要加密这个密码，请在命令行直接使用这个模块。</p> <pre>java org.picketbox. datasource.sec urity.SecureId entityLoginMod ule password_to_en crypt</pre> <p>将这个命令的运行结果粘贴到模块选项的 value 字段。默认值是一个空字符串。</p>
managedConnectionFactoryName	JCA 资源	无	数据源的 JCA 连接工厂的名称。

表 12.30. PropertiesUsers

Code	PropertiesUsers
类	org.jboss.security.auth.spi.PropertiesUsersLoginModule
描述	使用一个属性文件来存储用户名和密码。它没有提供授权（角色映射）。这个模块只适合于测试用途。

表 12.31. SimpleUsers

Code	SimpleUsers
类	org.jboss.security.auth.spi.SimpleUsersLoginModule
描述	这个登录模块使用 <i>module-option</i> 保存用户名和明文密码。 <i>module-option</i> 的 <i>name</i> 和 <i>value</i> 属性指定了用户名和密码。它只是用于测试目的，不适合用于产品环境里。

表 12.32. LdapUsers

Code	LdapUsers
------	------------------

类	<code>org.jboss.security.auth.spi.LdapUser sLoginModule</code>
描述	LdapUsers 模块被 ExtendedLDAP 和 AdvancedLdap 模块取代。

表 12.33. Kerberos

Code	Kerberos
类	<code>com.sun.security.auth.module.Krb5Log inModule</code>
描述	用 GSSAPI 执行Kerberos 登录验证。这个模块不是 Sun Microsystems 提供的 API 里的安全框架的一部分。细节可以在 http://docs.oracle.com/javase/7/docs/jre/api/security/jaas/spec/com/sun/security/auth/module/Krb5LoginModule.html 里找到。这个模块需要和另外一个处理验证和角色映射的模块配对。

表 12.34. Kerberos 模块选项

选项	类型	默认值	描述
storekey	true 或 false	false	是否添加 KerberosKey 到主题的私有凭证。
doNotPrompt	true 或 false	false	如果设置为 true ，用户将不会被提示输入密码。
useTicketCache	布尔值， true 或 false 。	false	如果为 true ，TGT 将从票据缓存里获取。如果为 false ，将不会使用票据缓存。
ticketcache	代表 Kerberos 票据缓存的文件或资源。	默认值取决于你所使用的操作系统。 <ul style="list-style-type: none"> 红帽企业版 Linux / Solaris: <code>/tmp/krb5cc_uid</code>，使用操作系统的数字 UID 值。 Microsoft Windows Server: 使用 Local Security Authority (LSA) API 来查找票据缓存。 	票据缓存的位置。

选项	类型	默认值	描述
useKeyTab	true 或 false	false	是否从密钥表文件里获取 principal 的密钥。
keytab	代表 Kerberos keytab 的文件或资源。	操作系统的 Kerberos 配置文件的位置，或者 /home/user/krb5.keytab 。	密钥表文件的位置。
principal	字符串	无	Principal 的名称。这可以是简单的用户名或服务名，如 host/testserver.acme.com 。或者当密钥表包含多个 principal 时，使用它而不是从密钥表里获取 principal 。
useFirstPass	true 或 false	false	是否以从 javax.security.auth.login.name 和 javax.security.auth.login.password 为关键字从模块的共享状态获取用户名和密码。如果验证失败，不会进行重试。
tryFirstPass	true 或 false	false	和 useFirstPass 相同，但如果验证失败，模块将使用 CallbackHandler 来获取新的用户名和密码。如果第二次验证失败，将报告给调用的应用程序。

选项	类型	默认值	描述
storePass	true 或 false	false	是否在模块的共享状态里保存用户名和密码。如果关键字已存在于共享内存里，或者验证失败的话，这都不会发生。
clearPass	true 或 false	false	设置它为 true 在两个验证阶段都完成后从共享内存里清除用户名和密码。

表 12.35. SPNEGO

Code	SPNEGO
类	org.jboss.security.negotiation.spnego.SPNEGOLoginModule
描述	允许在 Microsoft Active Directory 服务器或其他支持 SPNEGO 的环境里进行 SPNEGO 验证。SPNEGO 也可以包含 Kerberos 凭证。这个模块需要和另外一个处理验证和角色映射的模块配对。

表 12.36. SPNEGO 模块选项

选项	类型	默认值	描述
serverSecurityDomain	string	null	定义通过 kerberos 登录模块获取服务器服务的标识符所使用的域。这个属性是必须设置的。
removeRealmFromPrincipal	boolean	false	指定在进一步处理前，应该从主体删除 kerberos 区。
usernamePasswordDomain	string	null	指定当 Kerberos 验证失败时用作失效切换登录模块的其他安全域。

表 12.37. AdvancedLdap

Code	AdvancedLdap
类	org.jboss.security.negotiation.AdvancedLdapLoginModule
描述	提供额外功能的模块，如 SASL 和对 JAAS 安全域的使用。

表 12.38. AdvancedLdap 模块选项

选项	类型	默认值	描述
bindAuthentication	字符串	无	用于绑定到目录服务器的 SASL 验证的类型。
java.naming.provider.url	string	无	目录服务器的 URI。
baseCtxDN	全限定的 DN	无	要用作搜索基础的标识名。
baseFilter	代表 LDAP 搜索过滤器的字符串。	无	用于缩减搜索结果的过滤器。
roleAttributeID	代表 LDAP 属性的字符串值。	无	包含授权角色的名称的 LDAP 属性。
roleAttributeIsDN	true 或 false	false	这个角色属性是否是标识名 (Distinguished Name , DN) 。
roleNameAttributeID	代表 LDAP 属性的字符串。	无	包含实际角色属性的 RoleAttributeId 里所包含的属性。
recurseRoles	true 或 false	false	是否在 RoleAttributeId 里递归地搜索角色。
referralUserAttributeIDToCheck	attribute	无	如果您没有使用 Referral ，这个选项可被忽略。在使用 Referral 时，如果角色对象位于 Referral 内部，这个选项表示包含为某个角色（例如， member ）定义的用户属性名称。用户将根据这个属性名称的上下文进行检查。如果没有设置这个选项，检查总是会失败，所以角色对象无法存储在 Referral 树里。

表 12.39. AdvancedADLdap

Code	AdvancedADLdap
类	org.jboss.security.negotiation.AdvancedADLoginModule
描述	这个模块扩展了 AdvancedLdap 登录模块，并添加额外的和 Microsoft Active Directory 相关的参数。

表 12.40. UsersRoles

Code	UsersRoles
类	<code>org.jboss.security.auth.spi.UsersRolesLoginModule</code>
描述	支持存储在两个不同属性文件里的多个用户和角色的简单登录模块。

表 12.41. UsersRoles 模块选项

选项	类型	默认值	描述
usersProperties	文件或资源的路径。	users.properties	包含用户-密码映射的文件或资源。这个文件的格式是 username=password 。
rolesProperties	文件或资源的路径。	roles.properties	包含用户-角色映射的文件或资源。这个文件的格式是 username=role1,role2,role3 。
password-stacking	useFirstPass 或 false	false	useFirstPass 的值表示这个登录模块应该首先查看存储在 LoginContext 里关于这个标识符的信息。当堆积这个登录模块和其他模块时可以使用这个选项。
hashAlgorithm	代表密码的哈希算法的字符串。	none	用于 hash 密码的 java.security.MessageDigest 算法的名称。这个选项没有默认值，你必须显性地设置它来启用哈希算法。当指定了 hashAlgorithm 时， CallbackHandler 里包含的明文密码将在作为 inputPassword 参数传递给 UsernamePasswordLoginModule.validatePassword 前进行 hash。保存在 users.properties 文件里的密码必须进行同等的 hash。

选项	类型	默认值	描述
hashEncoding	base64 或 hex	base64	如果设置了 hashAlgorithm ，哈希密码的字符串格式。
hashCharset	字符串	容器的运行时环境里的默认编码集。	将明文密码转换为字节队列的编码。
unauthenticatedIdentity	主体名称	无	定义分配给不包含验证信息的请求的 principal 名称。这允许不受保护的 servlet 调用不要求专有角色的 EJB 上的方法。这样的 principal 没有关联的角色且只访问未设置安全性的 EJB 或者和 unchecked permission 约束关联的 EJB。

自定义验证模块

验证模块是 `javax.security.auth.spi.LoginModule` 的实现。关于创建自定义验证模块的更多信息，请参考相关的 API 文档。

[提交 bug 报告](#)

12.2. 包括的授权模块

下面的模块提供授权服务。

代码	类
DenyAll	<code>org.jboss.security.authorization.modules.AllDenyAuthorizationModule</code>
PermitAll	<code>org.jboss.security.authorization.modules.AllPermitAuthorizationModule</code>
Delegating	<code>org.jboss.security.authorization.modules.DelegatingAuthorizationModule</code>
Web	<code>org.jboss.security.authorization.modules.web.WebAuthorizationModule</code>
JACC	<code>org.jboss.security.authorization.modules.JACCAuthorizationModule</code>

代码	类
XACML	<code>org.jboss.security.authorization.modules.XACMLAuthorizationModule</code>

AllDenyAuthorizationModule

总是拒绝授权请求的简单授权模块。它没有可用的配置选项。

AllPermitAuthorizationModule

总是允许授权请求的简单授权模块。它没有可用的配置选项。

DelegatingAuthorizationModule

将决策制定委托给配置的委托者的默认授权模块。

WebAuthorizationModule

带有默认 Tomcat 授权逻辑（permit all）的默认 Web 授权模块。

JACCAuthorizationModule

这个模块用两个委托者（用于 Web 容器授权请求的 `WebJACCPolicyModuleDelegate` 和用于 EJB 容器请求的 `EJBJACCPolicyModuleDelegate`）来强制实施 JACC 模式。它没有可用的配置选项。

XACMLAuthorizationModule

这个模块用两个委托者（`WebXACMLPolicyModuleDelegate` 和 `EJBXACMLPolicyModuleDelegate`）对 Web 和 EJB 容器强制实施 XACML 授权。这个模块根据已注册的策略评估 Web 或 EJB 请求的结果来创建 PDP 对象。

AbstractAuthorizationModule

必须进行覆盖的基本授权模块，它提供委托至其他授权模块的功能。

[提交 bug 报告](#)

12.3. 所包括的安全映射模块

JBoss EAP 6 提供了下面的安全映射角色。

代码	类
PropertiesRoles	<code>org.jboss.security.mapping.providers.role.PropertiesRolesMappingProvider</code>
SimpleRoles	<code>org.jboss.security.mapping.providers.role.SimpleRolesMappingProvider</code>
DeploymentRoles	<code>org.jboss.security.mapping.providers.role.DeploymentRolesMappingProvider</code>
DatabaseRoles	<code>org.jboss.security.mapping.providers.role.DatabaseRolesMappingProvider</code>

代码	类
LdapRoles	<code>org.jboss.security.mapping.providers.role.LdapRolesMappingProvider</code>
LdapAttributes	<code>org.jboss.security.mapping.providers.attribute.LdapAttributeMappingProvider</code>

DeploymentRolesMappingProvider

负责可以在 `jboss-web.xml` 和 `jboss-app.xml` 部署描述符文件里完成的主体到角色映射的角色映射模块。

例 12.1. 示例

```
<jboss-web>
...
<security-role>
  <role-name>Support</role-name>
  <principal-name>Mark</principal-name>
  <principal-name>Tom</principal-name>
</security-role>
...
</jboss-web>
```

org.jboss.security.mapping.providers.DeploymentRoleToRolesMappingProvider

负责可以在 `jboss-web.xml` 和 `jboss-app.xml` 部署描述符文件里完成的主体到角色映射的角色到角色映射模块（Role to Roles Mapping Module）。在这个例子里，`principal-name` 表示映射其他角色的角色。

例 12.2. 示例

```
<jboss-web>
...
<security-role>
  <role-name>Employee</role-name>
  <principal-name>Support</principal-name>
  <principal-name>Sales</principal-name>
</security-role>
...
</jboss-web>
```

表示具有 Support 或 Sales 角色的个主体也会分配 Employee 角色。

org.jboss.security.mapping.providers.OptionsRoleMappingProvider

从选项里提取角色并附加到传入的组里的 **Role Mapping** 提供者。它使用角色名称（键）的属性风格映射和用逗号隔开的角色列表（值）。

org.jboss.security.mapping.providers.principal.SimplePrincipalMappingProvider

使用 **SimplePrincipal** 并转成具有不同主体名称的 **SimplePrincipal** 的主体映射提供者。

DatabaseRolesMappingProvider

从数据库读取角色的 **MappingProvider**。

选项：

- **dsJndiName**：用来将角色映射至用户的数据源的 JNDI 名称。
- **rolesQuery**：这个选项应该是等同于 "select RoleName from Roles where User=?" 的 prepared 语句，? 应用当前的主体名称替换。
- **suspendResume**：布尔值 - 在执行角色搜索时暂停且在之后恢复和当前线程关联的事务。
- **transactionManagerJndiName**：事务管理者的 JNDI 名称（默认为 java:/TransactionManager）

LdapRolesMappingProvider

为使用 LDAP 服务器搜索角色的用户分配角色的映射提供者。

选项：

- **bindDN**：用户和角色查询里用来绑定 LDAP 服务器的 DN。这个 DN 需要读取和搜索 **baseCtxDN** 和 **rolesCtxDN** 值上的权限。
- **bindCredential**：**bindDN** 的密码。如果指定了 **jaasSecurityDomain**，它可以被加密。
- **rolesCtxDN**：用于搜索用户角色的上下文的固定 DN。这不是实际角色的 DN，它是包含用户角色的对象所在的 DN。例如，在 Microsoft Active Directory 服务器里，它是用户帐号所在的 DN。
- **roleAttributeID**：包含授权角色的名称的 LDAP 属性。
- **roleAttributeIsDN**：**roleAttributeID** 是否包含角色对象的全限定 DN。如果为 **false**，角色名将从上下文名称的 **roleNameAttributeId** 属性值里获取。某些目录模式，如 Microsoft Active Directory，要求这个属性的值为 **true**。
- **roleNameAttributeID**：包含角色名称的 **roleCtxDN** 上下文里的属性的名称。如果 **roleAttributeIsDN** 属性为 **true**，这个属性将被用来查找角色对象的名属性。
- **parseRoleNameFromDN**：指定查询返回的 DN 是否包含 **roleNameAttributeID**。如果设置为 **true**，将检查 DN 里是否有 **roleNameAttributeID**，如果为 **false**，将不会检查。这个标记可以提高 LDAP 查询的性能。
- **roleFilter**：用来定位和验证用户相关联的角色的搜索过滤器。从登录模块回调方法里获得的输入用户名或 **userDN** 将被替换过滤器里的 **{0}** 表达式。已验证的 **userDN** 将替换过滤器里的 **{1}** 表达式。匹配输入用户名的搜索过滤器示例是 (**member={0}**)。对应已验证的用户 DN 的例子是 **userDN is (member={1})**。
- **roleRecursion**：对匹配内容进行角色搜索的递归层数。0 表示禁用递归。

- **searchTimeLimit** : 用户/角色搜索的超时时间 (毫秒) 。默认为 10000 (10 秒) 。
- **searchScope** : 所使用的搜索作用域。

PropertiesRolesMappingProvider

以下列格式从属性文件读取角色的 MappingProvider : username=role1,role2,...

选项 :

- **rolesProperties** : 属性格式文件的名称。JBoss 变量的扩展可以使用格式 `${jboss.variable}`。

SimpleRolesMappingProvider

从选项表里读取角色的简单 MappingProvider。Option 属性名称是分配角色的主体名称, 属性值是分配给主体的用逗号隔开的角色名称。

例 12.3. 示例

```
<module-option name="JavaDuke" value="JBossAdmin,Admin"/>
<module-option name="joe" value="Users"/>
```

org.jboss.security.mapping.providers.attribute.DefaultAttributeMappingProvider

检查模块并在映射上下文里定位主体名称以根据模块选项的 `principalName + ".email"` 创建属性电子邮件, 并将其映射至给定的主体。

LdapAttributeMappingProvider

将属性从 LDAP 映射至主题。它的选项包括 LDAP JNDI 供应商支持的任何选项。

例 12.4. 标准属性的名称示例包括 :

```
Context.INITIAL_CONTEXT_FACTORY = "java.naming.factory.initial"
Context.SECURITY_PROTOCOL = "java.naming.security.protocol"
Context.PROVIDER_URL = "java.naming.provider.url"
Context.SECURITY_AUTHENTICATION = "java.naming.security.authentication"
```

选项 :

- **bindDN** : 用户和角色查询里用来绑定 LDAP 服务器的 DN。这个 DN 需要读取和搜索 `baseCtxDN` 和 `rolesCtxDN` 值上的权限。
- **bindCredential** : `bindDN` 的密码。如果指定了 `jaasSecurityDomain`, 它可以被加密。
- **baseCtxDN** : 开始用户搜索所使用的上下文的固定标识名。
- **baseFilter** : 用来定位要验证的用户的上下文的搜索过滤器。从登录模块回调方法里获得的输入用户名或 `userDN` 将替换至过滤器里的 `{0}` 表达式。这种替换行为来自标准的 `__DirContext.search(Name, String, Object[], SearchControls cons)` 方法。搜索过滤器的一个常见例子是 `(uid={0})`。

- **searchTimeLimit** : 用户/角色搜索的超时时间 (毫秒) 。默认为 10000 (10 秒) 。
- **attributeList** : 用逗号隔开的属性列表。例如 , mail、cn、sn、employeeType、employeeNumber。
- **jaasSecurityDomain** : 用于解密 `java.naming.security.principal` 的 `JaasSecurityDomain`。 `JaasSecurityDomain#encrypt64(byte[])` 方法返回密码的解密形式。 `org.jboss.security.plugins.PBEUtils`也可以用来生成解密形式。

[提交 bug 报告](#)

12.4. 包括的安全审计供应商模块

JBoss EAP 提供了一个安全审计供应商。

代码	类
LogAuditProvider	org.jboss.security.audit.providers.LogAuditProvider

[提交 bug 报告](#)

第 13 章 子系统配置

13.1. 子系统配置概述

介绍

JBoss EAP 6 使用了简化的配置，对每个域或独立服务器使用一个配置文件。在域模式里也为每个主机控制器配备一个单独的文件。对配置的修改会自动持久化，所以您不要手动编辑 XML。管理 API 将自动扫描和覆盖配置。基于命令行的管理 CLI 和基于 Web 的管理控制台允许您配置 JBoss EAP 6 的每个方面。

JBoss EAP 6 构建在模块化类加载的概念上。平台提供的每个 API 或服务都实现为模块，再根据需要进行加载和卸载。大多数模块都包含一个配置元素，名为子系统 (**subsystem**)。子系统配置信息保存在统一的配置文件 **EAP_HOME/domain/configuration/domain.xml** (受管域) 或 **EAP_HOME/standalone/configuration/standalone.xml** (独立服务器) 里。许多子系统都包括通过之前 JBoss EAP 版本的部署描述符配置的配置细节。

子系统配置模式

每个子系统的配置都是在 XML Schema 里定义的。配置模式位于 JBoss 的 **EAP_HOME/docs/schema/** 目录里。

下面的子系统被称为 *simple subsystems*，因为它们没有任何可配置的属性或元素。它们通常列在配置文件的顶部。

简单子系统

- **ee**– Java EE 6 API 实现
- **ejb**– Enterprise JavaBeans (EJB) 子系统
- **jaxrs**– RESTeasy 提供的 JAX-RS API
- **sar**– 支持 Service Archives 的子系统
- **threads**– 支持进程线程的子系统
- **weld**– Weld 提供的上下文和依赖关系注入的 API

[提交 bug 报告](#)

第 14 章 日志子系统

14.1. 介绍

14.1.1. 日志概述

JBoss EAP 6 提供了高度可配置的日志功能以用于内部或应用程序。日志子系统基于 JBoss LogManager，它还支持 JBoss Logging 以外的几个第三方应用程序日志框架。

日志子系统是通过日志类别和处理程序体系来进行配置的。日志类别 (**Category**) 定义要记录哪些消息，而日志处理程序 (**Handler**) 则定义如何处理这些消息 (写入到磁盘、发送到控制台等)。

日志配置集 (**Logging Profile**) 允许创建日志配置的唯一名称并分配给不依赖其他日志配置的应用程序。日志配置集的配置和主日志子系统的配置基本上是一样的。

[提交 bug 报告](#)

14.1.2. JBoss LogManager 支持的应用程序日志框架

JBoss LogManager 支持下列日志框架：

- JBoss EAP 6 里包含的 JBoss Logging
- Apache Commons Logging - <http://commons.apache.org/logging/>
- Simple Logging Facade for Java (SLF4J) - <http://www.slf4j.org/>
- Apache log4j - <http://logging.apache.org/log4j/1.2/>
- Java SE Logging (java.util.logging) - <http://download.oracle.com/javase/6/docs/api/java/util/logging/package-summary.html>

[提交 bug 报告](#)

14.1.3. 配置引导日志

引导日志 (JBoss Logging) 记录服务器启动 (或"引导") 时发生的事件。

服务器启动时如果有可用的 **logging.properties** 文件，这些属性配置会用来记录日志子系统初始还前发生的事件。此后，日志子系统将接管事件的记录。

在您用管理 CLI 修改 **logging** 子系统或手动编辑服务器配置文件时，它会更新 **logging.properties** 文件。

如果安装时缺失了 **logging.properties**，在初始化日志子系统前引导过程中出现的任何日志消息都会丢失。而一旦初始化了日志子系统，消息又会出现于日志里。



警告

我们建议您不要直接编辑 **logging.properties** 文件，除非引导服务器出现严重问题时需要主机或进程控制器的额外日志信息。

[提交 bug 报告](#)

14.1.4. 关于垃圾收集日志

垃圾收集日志（Garbage Collection Logging）用普通文本日志文件记录所有的垃圾收集活动。这些日志文件可以用于诊断。从 JBoss EAP 6.3 开始，在所有被支持的配置里（除了 IBM JDK），对于 **standalone** 模式，垃圾收集日志默认都是启用的。

日志输出到 **\$EAP_HOME/standalone/log/gc.log.digit** 文件。日志轮换已经启用，日志文件的数量限制为 5 且每个文件最大为 3MB。

[提交 bug 报告](#)

14.1.5. 隐性的 Logging API 依赖关系

JBoss EAP 6 的日志子系统有一个控制容器是否添加隐性 Logging API 依赖关系到部署里的属性。这个属性默认被设置为 **true**，表示所有的隐性 Logging API 依赖关系都会被添加至部署里。如果设置为 **false**，隐性 Logging API 依赖关系就不会被添加。

add-logging-api-dependencies 属性可以用管理 CLI 进行配置。例如：

```
/subsystem=logging:write-attribute(name=add-logging-api-dependencies,
value=false)
```

[提交 bug 报告](#)

14.1.6. 默认的日志文件位置

这些日志文件是为默认的日志配置创建的。默认的配置使用定时日志处理程序写入到服务器日志文件里。

表 14.1. 独立服务器的默认日志文件

日志文件	描述
EAP_HOME/standalone/log/server.log	服务器日志包含了所有的服务器日志消息，其中包括服务器启动消息。
EAP_HOME/standalone/log/gc.log	垃圾收集日志。包含所有垃圾收集的细节。

表 14.2. 受管域的默认日志文件

日志文件	描述
EAP_HOME/domain/log/host-controller.log	主机控制器引导日志包含和主机控制器启动相关的日志消息。
EAP_HOME/domain/log/process-controller.log	进程控制器引导日志包含和进程控制器启动相关的日志消息。

日志文件	描述
EAP_HOME/domain/servers/SERVERNAME/log/server.log	名为 server. 的服务器日志包含了所有的服务器日志消息，其中包括服务器启动消息。

[提交 bug 报告](#)

14.1.7. 日志的过滤器表达式



注意

为 **root logger** 指定的 **filter-spec** 不是从其他处理程序继承的。您必须为每个处理程序指定一个 **filter-spec**。

表 14.3. 日志的过滤器表达式

过滤器类型 表达式	描述	参数
Accept accept	接受所有日志消息	accept
Deny deny	拒绝所有日志消息	deny
Not not[filter expression]	返回过滤器表达式的相反值	将单一过滤器表达式用作参数 not(match("JBAS"))
All all[filter expression]	返回来自多个过滤器表达式的串联值	使用多个用逗号隔开的过滤器表达式 all(match("JBAS"),match("WELD"))
Any any[filter expression]	返回一个来自多个过滤器表达式的值。	使用多个用逗号隔开的过滤器表达式 any(match("JBAS"),match("WELD"))
Level Change levelChange[level]	用指定的级别修改日志记录	以单个字符串级别为参数 levelChange("WARN")
Levels levels[levels]	用列表里的级别过滤日志消息	以用逗号隔开的基于多个字符串的级别为参数 levels("DEBUG","INFO","WARN","ERROR")

过滤器类型 表达式	描述	参数
<p>Level Range</p> <p><code>levelRange[minLevel,maxLevel]</code></p>	<p>过滤指定级别范围内的日志消息。</p>	<p>过滤器表达式使用 <code>[</code> 来指定最小包含级别以及用 <code>]</code> 指定最大包含级别。或者可以使用 <code>(</code> 或 <code>)</code> 来指定排除的级别。这个表达式的第一个参数是允许的最小级别，第二是允许的最大级别。</p> <p>下面是一些例子。</p> <ul style="list-style-type: none"><code>levelRange("DEBUG","ERROR")</code> 最小级别必须大于 DEBUG，而最大级别必须小于 ERROR。<code>levelRange["DEBUG","ERROR"]</code> 最小级别必须大于或等于 DEBUG，而最大级别必须小于 ERROR。<code>levelRange["INFO","ERROR"]</code> 最小级别必须大于或等于 INFO，而最大级别必须小于或等于 ERROR。
<p><code>Match(match["pattern"])</code></p>	<p>基于常规表达式的过滤器。根据表达式里的指定的模式使用未格式化的消息。</p>	<p>以常规表达式为参数</p> <p><code>match("JBAS\d+")</code></p>
<p>Substitute</p> <p><code>(substitute["pattern","replacement value"])</code></p>	<p>这个过滤器用文本替换第一个匹配的模式</p>	<p>这个表达式的第一个参数是模式，而第二个参数是替换文本</p> <p><code>substitute("JBAS","EAP")</code></p>
<p>Substitute All</p> <p><code>(substituteAll["pattern","replacement value"])</code></p>	<p>这个过滤器用文本替换所有匹配的模式</p>	<p>这个表达式的第一个参数是模式，而第二个参数是替换文本</p> <p><code>substituteAll("JBAS","EAP")</code></p>

[提交 bug 报告](#)

14.1.8. 关于日志级别

日志级别是一系列排序的枚举值，表示日志消息的性质和严重性。给定日志消息的级别是开发人员用所选的日志框里合适的方法指定的。

JBoss EAP 6 支持受支持的应用日志框架使用的所有日志级别。最常用的 6 个日志级别是（从低到高）：**TRACE**、**DEBUG**、**INFO**、**WARN**、**ERROR** 和 **FATAL**。

日志类别和处理程序使用日志级别来限制它们负责的日志消息。每个日志级别都分配有一个数值，表示相对于其他日志级别的顺序。日志类别和处理程序分配的日志级别限制它们只成立了该级别或更高级别的消息。例如，级别为 **WARN** 的日志处理程序只记录级别为 **WARN**、**ERROR** 和 **FATAL** 的日志消息。

[提交 bug 报告](#)

14.1.9. 支持的日志级别

表 14.4. 支持的日志级别

日志级别	值	描述
FINEST	300	-
FINER	400	-
TRACE	400	用于提供应用程序运行状态详细信息的信息。 TRACE 级别的信息通常在调试应用程序时被捕获。
DEBUG	500	用于指明应用程序的单独请求或活动进度的信息。 TRACE 级别的信息通常在调试应用程序时被捕获。
FINE	500	-
CONFIG	700	-
INFO	800	用于表示应用程序总体进度的信息。通常用于应用程序启动、关闭或其他主要的生命周期事件。
WARN	900	用于表示未发生错误但并不理想的情形。它可以表示某些在将来可能导致错误的情形。
WARNING	900	-
ERROR	1000	用于表示已经发生并会阻止当前活动或请求的完成、但不会阻止应用程序运行的错误。
SEVERE	1000	-
FATAL	1100	用于表示可能导致严重服务故障和应用程序关闭及 JBoss EAP 6 关闭的事件。

[提交 bug 报告](#)

14.1.10. 关于日志类别

日志类别定义一系列要捕获的日志消息以及处理这些消息的处理程序。

要记录的日志消息是通过它们的原始 **Java** 软件包和日志级别来定义的。这些软件包里类的消息以及该日志级别或更低级别的消息将按日志类别记录并发送到指定的日志处理程序。

日志类别可以选择使用 **Root Logger** 的日志处理程序而不是自己的处理程序。

[提交 bug 报告](#)

14.1.11. 关于 Root Logger

Root logger 将记录所有发送到服务器而未按照日志列别记录的日志消息（指定级别）。这些消息会被发送到一个或多个日志处理程序。

在默认情况下，我们配置 **Root Logger** 使用控制台和定期日志处理程序。定期日志处理程序写入文件 **server.log**。这个文件有时候被称为服务器日志（**Server Log**）。

[提交 bug 报告](#)

14.1.12. 关于日志处理程序

日志处理程序（**Log Handler**）定义 **JBoss EAP 6** 如何记录捕获的日志消息。您可以配置 6 种处理程序类型：**Console**、**File**、**Periodic**、**Size**、**Async** 和 **Custom**。

[提交 bug 报告](#)

14.1.13. 日志处理程序的类型

Console

Console 处理程序写入日志消息到主机操作系统的标准输出流（**stdout**）或标准错误流（**stderr**）。当通过命令行提示运行 **JBoss EAP 6** 时会显示这些消息。除非配置操作系统捕获标准输出和标准错误流，来自 **Console** 日志处理程序的消息不会被保存。

File

File 日志处理程序是最简单的处理程序，它将日志消息写入到指定文件里。

Periodic

Periodic 日志处理程序将日志消息写入到命名文件，直到指定的时间过期。一旦超过指定的时间，文件将通过附件指定的时间戳来改名，而处理程序继续写入到根据原来名称新创建的日志文件。

Size

Size 日志处理程序将日志消息写入到命名文件，直至到达指定的文件大小。一旦超过指定的大小，文件将通过数字前缀改名，而处理程序继续写入到根据原来名称新创建的日志文件。每个 **Size** 日志处理程序必须指定以这种方式保存的文件的最大数目。

Async

Async 日志处理程序是为一个或多个其他日志处理程序提供异步行为的 **Wrapper** 日志处理程序。这对于有高延迟性或其他性能问题的日志处理程序来说很有用，例如写入日志文件到网络文件系统。

Custom

Custom 日志处理程序让您可以配置新的日志处理程序类型。**Custom** 日志处理程序必须实现为继承 **java.util.logging.Handler** 的 **Java** 类，并包含在模块里。

syslog

Syslog 处理程序用于发送消息到远程日志服务器。它允许多个应用程序发送它们的日志消息到相同的服务器，一起进行解析。

[提交 bug 报告](#)

14.1.14. 关于日志格式器

日志格式器 (Log Formatter) 是日志处理程序的配置属性，它定义日志消息的外观。它是基于 `java.util.Formatter` 类的语法的子字符串。

例如 `default` 配置里的日志格式器 `%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n`，它创建这样的日志消息：

```
15:53:26,546 INFO [org.jboss.as] (Controller Boot Thread) JBAS015951:
Admin console listening on http://127.0.0.1:9990
```

[提交 bug 报告](#)

14.1.15. 日志格式器语法

表 14.5. 日志格式器语法

符号	描述
<code>%c</code>	日志事件的类别
<code>%p</code>	日志条目的级别 (INFO/DEBUG/etc)
<code>%P</code>	日志条目的地区级别
<code>%d</code>	当前的日期/时间：date/time (yyyy-MM-dd HH:mm:ss,SSS form)
<code>%r</code>	相对时间 (日志被初始化起的毫秒数)
<code>%z</code>	时区
<code>%k</code>	日志资源关键字 (用于日志消息的本地化)
<code>%m</code>	日志消息 (包含关于异常的跟踪信息)
<code>%s</code>	简单的日志消息 (无异常跟踪信息)
<code>%e</code>	异常跟踪信息 (无扩展模块信息)
<code>%E</code>	异常跟踪信息 (有扩展模块信息)
<code>%t</code>	当前线程的名称

符号	描述
%n	新行符号
%C	调用日志方法（慢速）的代码的类
%F	调用日志方法（慢速）的代码的文件名
%l	调用日志方法（慢速）的代码的位置
%L	调用日志方法（慢速）的代码的行号
%M	调用日志方法（慢速）的代码的方法
%x	嵌套的诊断上下文
%X	消息诊断上下文
%%	百分比（已完成）

[提交 bug 报告](#)

14.2. 在管理控制台里配置日志

管理控制台为 **Root Logger**、日志处理程序、日志类别的配置提供了图形化用户界面。您可以通过下列步骤在管理控制台里找到日志配置：

您可用下列步骤访问这个配置：

1. 登录到管理控制台
2. 进入日志子系统配置。根据服务器是运行于受管域还是独立服务器模式，这个步骤会有所不同。
 - **独立服务器**
点击 **Configuration**，展开 **Subsystems** 菜单里的 **Core**，然后点击 **Logging**。
 - **受管域**
点击 **Configuration**，从下拉菜单里选择要编辑的配置集。展开 **Subsystems** 菜单里的 **Core**，然后点击 **Logging**。

您可以配置 **Root Logger** 的任务是：

- 编辑日志级别。
- 添加和删除日志处理程序。

您可以配置日志类别的任务是；

- 添加和删除日志级别。
- 编辑日志类别属性。

- 从列表里添加和删除日志处理程序。

配置日志处理程序的主要任务是：

- 添加新的处理程序。
- 配置处理程序。

所有被支持的 6 个日志处理程序（包括 Custom）都可以在管理控制台里进行配置。

[提交 bug 报告](#)

14.3. CLI 里的日志配置

前提条件

管理 CLI 必须在运行并连接至相关的 JBoss EAP 实例。关于进一步的信息，请参考 [第 3.5.2 节“启动管理 CLI”](#)。

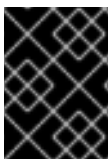
[提交 bug 报告](#)

14.3.1. 用 CLI 配置 Root Logger

Root Logger 的配置可以通过 CLI 查看和编辑。

您可以配置 Root Logger 的主要任务是：

- 在 Root Logger 里添加日志处理程序。
- 显示 Root Logger 配置。
- 改变日志级别。
- 在 Root Logger 里删除日志处理程序。



重要

当在日志配置集里配置 Root Logger 时，配置路径的根目录是 `/subsystem=logging/logging-profile=NAME/` 而不是 `/subsystem=logging/`。

在 Root Logger 里添加日志处理程序。

请使用 `add-handler` 操作和下列语法，这里的 *HANDLER* 是要添加的日志处理程序的名称。

```
/subsystem=logging/root-logger=ROOT:add-handler(name="HANDLER")
```

这个日志处理程序必须在添加到 Root Logger 之前必须已被创建。

例 14.1. Root Logger 的 add-handler 操作

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:add-handler(name="FILE")
{"outcome" => "success"}
```

显示 Root Logger 配置的内容。

使用 **read-resource** 操作和下列语法。

```
/subsystem=logging/root-logger=R00T:read-resource
```

例 14.2. Root Logger 的 read-resource 操作

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=R00T:read-resource
{
  "outcome" => "success",
  "result" => {
    "filter" => undefined,
    "filter-spec" => undefined,
    "handlers" => [
      "CONSOLE",
      "FILE"
    ],
    "level" => "INFO"
  }
}
```

设置 Root Logger 的日志级别。

请使用 **write-attribute** 命令和下列语法，这里的 *LEVEL* 是被支持的日志级别之一。

```
/subsystem=logging/root-logger=R00T:write-attribute(name="level",
value="LEVEL")
```

例 14.3. 使用 Root Logger 的 write-attribute 操作来设置日志级别

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=R00T:write-attribute(name="level", value="DEBUG")
{"outcome" => "success"}
```

从 Root Logger 删除日志处理程序。

请使用 **remove-handler** 操作和下列语法，这里的 *HANDLER* 是要删除的日志处理程序的名称。

```
/subsystem=logging/root-logger=R00T:remove-handler(name="HANDLER")
```

例 14.4. 删除日志处理程序

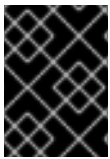
```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=R00T:remove-handler(name="FILE")
{"outcome" => "success"}
```

14.3.2. 在 CLI 里配置日志类别

您可以在 CLI 里添加、删除和编辑日志类别。

配置日志类别的主要任务是：

- 添加新的日志类别。
- 显示日志类别的配置。
- 设置日志级别。
- 在日志类别里添加日志程序。
- 在日志类别里删除日志处理程序。
- 删除日志类别。



重要

当在日志配置集里配置日志类别时，配置路径的根目录是 `/subsystem=logging/logging-profile=NAME/` 而不是 `/subsystem=logging/`。

添加日志类别

请使用 **write-attribute** 命令和下列语法。用日志类别的名称替换 *CATEGORY*，并用日志级别替换 *LEVEL*。

```
/subsystem=logging/logger=CATEGORY:add
```

例 14.5. 添加新的日志类别

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:add
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

显示日志类别配置

请使用 **read-resource** 命令和下列语法。用日志类别的名称替换 *CATEGORY*。

```
/subsystem=logging/logger=CATEGORY:read-resource
```

例 14.6. 日志类别的 read-resource 操作

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=org.apache.tomcat.util.modeler:read-
resource
{
    "outcome" => "success",
    "result" => {
        "category" => "org.apache.tomcat.util.modeler",
        "filter" => undefined,
```

```

        "filter-spec" => undefined,
        "handlers" => undefined,
        "level" => "WARN",
        "use-parent-handlers" => true
    }
}
[standalone@localhost:9999 /]

```

设置日志级别

请使用 **write-attribute** 命令和下列语法。用日志类别的名称替换 *CATEGORY*，并用日志级别替换 *LEVEL*。

```

/subsystem=logging/logger=CATEGORY:write-attribute(name="level",
value="LEVEL")

```

例 14.7. 设置日志级别

```

[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:write-
attribute(name="level", value="DEBUG")
{"outcome" => "success"}
[standalone@localhost:9999 /]

```

设置日志类别以使用 Root Logger 日志处理程序。

请使用 **write-attribute** 命令和下列语法。用日志类别的名称替换 *CATEGORY*。根据这个日志类别是否使用 Root Logger 处理程序用 **true** 替换 *BOOLEAN*。如果只使用自己分配的处理程序，则用 **false** 来代替。

```

/subsystem=logging/logger=CATEGORY:write-attribute(name="use-parent-
handlers", value="BOOLEAN")

```

例 14.8. 设置 use-parent-handlers

```

[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:write-
attribute(name="use-parent-handlers", value="true")
{"outcome" => "success"}
[standalone@localhost:9999 /]

```

在日志类别里添加日志程序

请使用 **add-handler** 命令和下列语法。用日志类别的名称替换 *CATEGORY*，并用要添加的处理程序的名称替换 *HANDLER*。

```

/subsystem=logging/logger=CATEGORY:add-handler(name="HANDLER")

```

这个日志处理程序必须在添加到 Root Logger 之前必须已被创建。

例 14.9. 添加日志处理程序

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:add-
handler(name="AccountsNFSAsync")
{"outcome" => "success"}
```

在日志类别里删除日志处理程序

请使用 **remove-handler** 命令和下列语法。用日志类别的名称替换 *CATEGORY*，并用要删除的处理程序的名称替换 *HANDLER*。

```
/subsystem=logging/logger=CATEGORY:remove-handler(name="HANDLER")
```

例 14.10. 删除日志处理程序

```
[standalone@localhost:9999 /] /subsystem=logging/logger=jacorb:remove-
handler(name="AccountsNFSAsync")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

删除类别

请使用 **remove** 命令和下列语法。用要删除的日志类别的名称替换 *CATEGORY*。

```
/subsystem=logging/logger=CATEGORY:remove
```

例 14.11. 删除日志类别

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[提交 bug 报告](#)**14.3.3. 在 CLI 里配置控制台日志处理程序**

您可以在 CLI 里添加、删除和编辑日志处理程序。

配置控制台日志处理程序的主要任务是：

- 添加新的控制台处理程序
- 显示控制台日志处理程序的配置。
- 设置处理程序的日志级别。
- 设置用于处理程序输出的目标。

- 设置用于处理程序输出的编码。
- 设置用于处理程序输出的格式器。
- 设置处理程序是否使用自动冲刷。
- 删除控制台日志处理程序。



重要

当在日志配置集里配置日志处理程序时，配置路径的根目录是 `/subsystem=logging/logging-profile=NAME/` 而不是 `/subsystem=logging/`。

添加控制台处理程序

请使用 **add** 操作和下列语法，用要添加的日志处理程序的名称替换 *HANDLER*。

```
/subsystem=logging/console-handler=HANDLER:add
```

例 14.12. 添加控制台处理程序

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:add
{"outcome" => "success"}
```

显示控制台日志处理程序的配置

请使用 **read-resource** 命令和下列语法。用日志处理程序的名称替换 *HANDLER*。

```
/subsystem=logging/console-handler=HANDLER:read-resource
```

例 14.13. 显示控制台日志处理程序的配置

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=CONSOLE:read-resource
{
  "outcome" => "success",
  "result" => {
    "autoflush" => true,
    "enabled" => true,
    "encoding" => undefined,
    "filter" => undefined,
    "filter-spec" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => "INFO",
    "name" => "CONSOLE",
    "named-formatter" => "COLOR-PATTERN",
    "target" => "System.out"
  }
}
```

设置日志级别

请使用 **write-attribute** 命令和下列语法。用控制台日志处理程序的名称替换 *HANDLER*，并用日志级别替换 *LEVEL*。

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="level", value="INFO")
```

例 14.14. 设置日志级别

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-attribute(name="level", value="TRACE") {"outcome" => "success"}
```

设置目标

请使用 **write-attribute** 命令和下列语法。用控制台日志处理程序的名称替换 *HANDLER*，用分别代表标准错误流和标准输出流的 **System.err** 或 **System.out** 替换 *TARGET*。

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="target", value="TARGET")
```

例 14.15. 设置目标

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-attribute(name="target", value="System.err") {"outcome" => "success"}
```

设置编码

请使用 **write-attribute** 命令和下列语法。用控制台日志处理程序的名称替换 *HANDLER*，并用所需的字符编码系统替换 *ENCODING*。

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="encoding", value="ENCODING")
```

例 14.16. 设置编码

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-attribute(name="encoding", value="utf-8") {"outcome" => "success"}
```

设置格式器

请使用 **write-attribute** 命令和下列语法。用控制台日志处理程序的名称替换 *HANDLER*，并用格式器字符串替换 *FORMAT*。

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="formatter", value="FORMAT")
```

例 14.17. 设置格式器

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:write-attribute(name="formatter",
value="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n")
{"outcome" => "success"}
```

设置自动冲刷

请使用 **write-attribute** 命令和下列语法。用控制台日志处理程序的名称替换 *HANDLER*。如果处理程序立即写入到输出，则用 **true** 替换 *BOOLEAN*。

```
/subsystem=logging/console-handler=HANDLER:write-
attribute(name="autoflush", value="BOOLEAN")
```

例 14.18. 设置自动冲刷

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:write-attribute(name="autoflush", value="true")
{"outcome" => "success"}
```

删除控制台日志处理程序

请使用 **remove** 命令和下列语法。用要删除的日志处理程序的名称替换 *HANDLER*。

```
/subsystem=logging/console-handler=HANDLER:remove
```

例 14.19. 删除控制台日志处理程序

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:remove
{"outcome" => "success"}
```

[提交 bug 报告](#)

14.3.4. 在 CLI 里配置文件日志处理程序

您可以在 CLI 里添加、删除和编辑文件日志处理程序。

配置文件日志处理程序的主要任务是：

- 添加新的文件处理程序。
- 显示文件日志处理程序的配置
- 设置处理程序的日志级别。
- 设置处理程序的附加行为。

- 设置处理程序是否使用自动冲刷。
- 设置用于处理程序输出的编码。
- 指定日志处理程序将写入的文件。
- 设置用于处理程序输出的格式器。
- 删除文件日志处理程序。



重要

当在日志配置集里配置日志处理程序时，配置路径的根目录是 `/subsystem=logging/logging-profile=NAME/` 而不是 `/subsystem=logging/`。

添加文件处理程序

请使用 **add** 命令和下列语法。用写入的日志文件的名称替换 *PATH*。用文件所在的目录的名称替换 *DIR*。*DIR* 的值可以是一个路径变量。

```
/subsystem=logging/file-handler=HANDLER:add(file={"path"=>"PATH",
"relative-to"=>"DIR"})
```

例 14.20. 添加文件处理程序

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:add(file={"path"=>"accounts.log", "relative-
to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

显示文件日志处理程序的配置

请使用 **read-resource** 命令和下列语法。用文件日志处理程序的名称替换 *HANDLER*。

```
/subsystem=logging/file-handler=HANDLER:read-resource
```

例 14.21. 使用 read-resource 操作

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:read-resource
{
  "outcome" => "success",
  "result" => {
    "append" => true,
    "autoflush" => true,
    "encoding" => undefined,
    "file" => {
      "path" => "accounts.log",
      "relative-to" => "jboss.server.log.dir"
    },
    "filter" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E\n",
```

```

        "level" => undefined
    }
}

```

设置日志级别

请使用 **write-attribute** 命令和下列语法。用文件日志处理程序的名称替换 *HANDLER*，并用日志级别替换 *LOG_LEVEL_VALUE*。

```

/subsystem=logging/file-handler=HANDLER:write-attribute(name="level",
value="LOG_LEVEL_VALUE")

```

例 14.22. 改变日志级别

```

/subsystem=logging/file-handler=accounts_log:write-
attribute(name="level", value="DEBUG")
{"outcome" => "success"}
[standalone@localhost:9999 /]

```

设置附加行为

请使用 **write-attribute** 命令和下列语法。用文件日志处理程序的名称替换 *HANDLER*。如果要求每次启动服务器时都创建新的日志文件，则用 **false** 替换 *BOOLEAN*。如果应用服务器应该继续使用相同的文件，则请用 **true** 替换 *BOOLEAN*。

```

/subsystem=logging/file-handler=HANDLER:write-attribute(name="append",
value="BOOLEAN")

```

例 14.23. 修改附加的属性

```

[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:write-attribute(name="append", value="true")
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
[standalone@localhost:9999 /]

```

重启 JBoss EAP 6 服务器以使修改生效。

设置自动冲刷

请使用 **write-attribute** 命令和下列语法。用文件日志处理程序的名称替换 *HANDLER*。如果处理程序立即写入到输出，则用 **true** 替换 *BOOLEAN*。

```

/subsystem=logging/file-handler=HANDLER:write-
attribute(name="autoflush", value="BOOLEAN")

```

例 14.24. 设置自动冲刷属性

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:write-attribute(name="autoflush", value="false")
{
    "outcome" => "success",
    "response-headers" => {"process-state" => "reload-required"}
}
[standalone@localhost:9999 /]
```

重启 JBoss EAP 6 服务器以使修改生效。

设置编码

请使用 **write-attribute** 命令和下列语法。用文件日志处理程序的名称替换 *HANDLER*，并用所需的字符编码系统替换 *ENCODING*。

```
/subsystem=logging/file-handler=HANDLER:write-
attribute(name="encoding", value="ENCODING")
```

例 14.25. 设置编码

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:write-attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
```

指定日志处理程序将写入的文件

请使用 **write-attribute** 命令和下列语法。用写入的日志文件的名称替换 *PATH*。用文件所在的目录的名称替换 *DIR*。*DIR* 的值可以是一个路径变量。

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="file",
value={"path"=>"PATH", "relative-to"=>"DIR"})
```

例 14.26. 指定日志处理程序将写入的文件

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:write-attribute(name="file", value=
{"path"=>"accounts-debug.log", "relative-
to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

设置格式器

请使用 **write-attribute** 命令和下列语法。用文件日志处理程序的名称替换 *HANDLER*，并用格式器字符串替换 *FORMAT*。

```
/subsystem=logging/file-handler=HANDLER:write-
attribute(name="formatter", value="FORMAT")
```

例 14.27. 设置格式器

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts-log:write-attribute(name="formatter",
value="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

删除文件日志处理程序

请使用 **remove** 命令和下列语法。用要删除的日志处理程序的名称替换 *HANDLER*。

```
/subsystem=logging/file-handler=HANDLER:remove
```

例 14.28. 删除文件日志处理程序

```
[standalone@localhost:9999 /] /subsystem=logging/file-
handler=accounts_log:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

日志处理程序只有没有被本地类别或异步日志处理程序引用时才可以被删除。

[提交 bug 报告](#)**14.3.5. 在 CLI 里配置定期日志处理程序**

您可以在 CLI 里添加、删除和编辑定期日志处理程序。

配置定期日志处理程序的主要任务是：

- 添加新的定期日志处理程序。
- 显示定期日志处理程序的配置
- 设置处理程序的日志级别。
- 设置处理程序的附加行为。
- 设置处理程序是否使用自动冲刷。
- 设置用于处理程序输出的编码。
- 指定日志处理程序将写入的文件。
- 设置用于处理程序输出的格式器。
- 设置用于轮换日志的后缀
- 删除定期日志处理程序。

每个任务都将在下面进行描述。



重要

当在日志配置集里配置日志处理程序时，配置路径的根目录是 `/subsystem=logging/logging-profile=NAME/` 而不是 `/subsystem=logging/`。

添加新的定期轮换文件日志处理程序

使用 **add** 操作和下列语法。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:add(file={
  "path"=>"PATH", "relative-to"=>"DIR"}, suffix="SUFFIX")
```

用日志文件的名称替换 *HANDLER*。用写入的日志文件的名称替换 *PATH*。用文件所在的目录的名称替换 *DIR*。*DIR* 的值可以是一个路径变量。用轮换后缀替换 *SUFFIX*。

例 14.29. 添加新的日志处理程序

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-
file-handler=HOURLY_DEBUG:add(file={"path"=>"daily-debug.log",
"relative-to"=>"jboss.server.log.dir"}, suffix=".yyyy.MM.dd")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

显示定期轮换文件日志处理程序的配置

使用 **read-resource** 操作和下列语法。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:read-resource
```

用文件日志处理程序的名称替换 *HANDLER*。

例 14.30. 使用 read-resource 操作

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-
file-handler=HOURLY_DEBUG:read-resource
{
  "outcome" => "success",
  "result" => {
    "append" => true,
    "autoflush" => true,
    "encoding" => undefined,
    "file" => {
      "path" => "daily-debug.log",
      "relative-to" => "jboss.server.log.dir"
    },
    "filter" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => undefined
  }
}
```


设置日志级别

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="level", value="LOG_LEVEL_VALUE")
```

用定期日志处理程序的名称替换 *HANDLER*，并用日志级别替换 *LOG_LEVEL_VALUE*。

例 14.31. 设置日志级别

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-
file-handler=HOURLY_DEBUG:write-attribute(name="level",
value="DEBUG")
{"outcome" => "success"}
```

设置附加行为

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/periodic-rotating-handler=HANDLER:write-attribute(name="append", value="BOOLEAN")
```

用定期日志处理程序的名称替换 *HANDLER*。如果要求每次启动服务器时都创建新的日志文件，则用 **false** 替换 *BOOLEAN*。如果应用服务器应该继续使用相同的文件，则请用 **true** 替换 *BOOLEAN*。

重启 JBoss EAP 6 服务器以使修改生效。

例 14.32. 设置附加行为

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-
file-handler=HOURLY_DEBUG:write-attribute(name="append", value="true")
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
```

设置自动冲刷

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="autoflush", value="BOOLEAN")
```

用定期日志处理程序的名称替换 *HANDLER*。如果处理程序立即写入到输出，则用 **true** 替换 *BOOLEAN*。

重启 JBoss EAP 6 服务器以使修改生效。

例 14.33. 设置自动冲刷行为

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-
file-handler=HOURLY_DEBUG:write-attribute(name="autoflush",
value="false")
{
    "outcome" => "success",
    "response-headers" => {"process-state" => "reload-required"}
}
```

设置编码

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-
attribute(name="encoding", value="ENCODING")
```

用定期日志处理程序的名称替换 *HANDLER*，并用所需的字符编码系统替换 *ENCODING*。

例 14.34. 设置编码

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-
file-handler=HOURLY_DEBUG:write-attribute(name="encoding", value="utf-
8")
{"outcome" => "success"}
```

指定日志处理程序将写入的文件

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-
attribute(name="file", value={"path"=>"PATH", "relative-to"=>"DIR"})
```

用定期文件的名称替换 *HANDLER*。用写入的日志文件的名称替换 *PATH*。用文件所在的目录的名称替换 *DIR*。*DIR* 的值可以是一个路径变量。

例 14.35. 指定日志处理程序将写入的文件

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-
file-handler=HOURLY_DEBUG:write-attribute(name="file", value=
{"path"=>"daily-debug.log", "relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```

设置格式器

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="formatter", value="FORMAT")
```

用定期日志处理程序的名称替换 *HANDLER*，并用格式器字符串替换 *FORMAT*。

例 14.36. 设置格式器

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-handler=HOURLY_DEBUG:write-attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n") {"outcome" => "success"} [standalone@localhost:9999 /]
```

设置用于轮换日志的后缀

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="suffix", value="SUFFIX")
```

用日志处理程序的名称替换 *HANDLER*，并用所需的后缀替换 *SUFFIX*。

例 14.37.

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-handler=HOURLY_DEBUG:write-attribute(name="suffix", value=".yyyy-MM-dd-HH") {"outcome" => "success"} [standalone@localhost:9999 /]
```

删除定期日志处理程序

使用 **remove** 操作和下列语法。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:remove
```

用定期日志处理程序的名称替换 *HANDLER*。

例 14.38. 删除定期日志处理程序

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-handler=HOURLY_DEBUG:remove {"outcome" => "success"} [standalone@localhost:9999 /]
```

[提交 bug 报告](#)

14.3.6. 在 CLI 里配置 Size 日志处理程序

您可以在 CLI 里添加、删除和编辑 **Size** 轮换文件日志处理程序。

配置 **Size** 日志处理程序的主要任务是：

- 添加新的日志处理程序。
- 显示日志处理程序的配置
- 设置处理程序的日志级别。
- 设置处理程序的附加行为。
- 设置处理程序是否使用自动冲刷。
- 设置用于处理程序输出的编码。
- 指定日志处理程序将写入的文件。
- 设置用于处理程序输出的格式器。
- 设置每个日志文件的最大尺寸。
- 设置要保留的备份日志的最大数目。
- 为 **Size** 轮换文件处理程序设置 **rotate on boot** 选项
- 删除日志处理程序。

每个任务都将在下面进行描述。



重要

当在日志配置集里配置日志处理程序时，配置路径的根目录是 **/subsystem=logging/logging-profile=NAME/** 而不是 **/subsystem=logging/**。

添加新的日志处理程序。

使用 **add** 操作和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:add(file={
"path"=>"PATH", "relative-to"=>"DIR"})
```

用日志文件的名称替换 **HANDLER**。用写入的日志文件的名称替换 **PATH**。用文件所在的目录的名称替换 **DIR**。**DIR** 的值可以是一个路径变量。

例 14.39. 添加新的日志处理程序。

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:add(file={"path"=>"accounts_trace.log",
"relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```

显示日志处理程序的配置

使用 **read-resource** 操作和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:read-resource
```

用文件日志处理程序的名称替换 *HANDLER*。

例 14.40. 显示日志处理程序的配置

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:read-resource
{
    "outcome" => "success",
    "result" => {
        "append" => true,
        "autoflush" => true,
        "encoding" => undefined,
        "file" => {
            "path" => "accounts_trace.log",
            "relative-to" => "jboss.server.log.dir"
        },
        "filter" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => undefined,
        "max-backup-index" => 1,
        "rotate-size" => "2m"
    }
}
[standalone@localhost:9999 /]
```

设置处理程序的日志级别

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="level", value="LOG_LEVEL_VALUE")
```

用日志处理程序的名称替换 *HANDLER*，并用日志级别替换 *LOG_LEVEL_VALUE*。

例 14.41. 设置处理程序的日志级别

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="level", value="TRACE")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

设置处理程序的附加行为。

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="append", value="BOOLEAN")
```

用文件日志处理程序的名称替换 *HANDLER*。如果要求每次启动服务器时都创建新的日志文件，则用 **false** 替换 *BOOLEAN*。如果应用服务器应该继续使用相同的文件，则请用 **true** 替换 *BOOLEAN*。

重启 JBoss EAP 6 服务器以使修改生效。

例 14.42. 设置处理程序的附加行为。

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="append", value="true")
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
[standalone@localhost:9999 /]
```

设置处理程序是否使用自动冲刷。

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="autoflush", value="BOOLEAN")
```

用文件日志处理程序的名称替换 *HANDLER*。如果处理程序立即写入到输出，则用 **true** 替换 *BOOLEAN*。

例 14.43. 设置处理程序是否使用自动冲刷。

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="autoflush", value="true")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

设置用于处理程序输出的编码。

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="encoding", value="ENCODING")
```

用文件日志处理程序的名称替换 *HANDLER*，并用所需的字符编码系统替换 *ENCODING*。

例 14.44. 设置用于处理程序输出的编码。

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
```

指定日志处理程序将写入的文件。

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="file", value={"path"=>"PATH", "relative-to"=>"DIR"})
```

用日志文件的名称替换 *HANDLER*。用写入的日志文件的名称替换 *PATH*。用文件所在的目录的名称替换 *DIR*。*DIR* 的值可以是一个路径变量。

例 14.45. 指定日志处理程序将写入的文件。

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-handler=ACCOUNTS_TRACE:write-attribute(name="file", value={"path"=>"accounts_trace.log", "relative-to"=>"jboss.server.log.dir"}, {"outcome" => "success"})
```

设置用于处理程序输出的格式器。

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="formatter", value="FORMATTER")
```

用文件日志处理程序的名称替换 *HANDLER*，并用格式器字符串替换 *FORMAT*。

例 14.46. 设置用于处理程序输出的格式器。

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-handler=ACCOUNTS_TRACE:write-attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p (%c) [%t] %s%E%n"), {"outcome" => "success"})
```

设置每个日志文件的最大尺寸。

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="rotate-size", value="SIZE")
```

用文件日志处理程序的名称替换 *HANDLER*，并用文件大小的最大值替换 *SIZE*。

例 14.47. 设置每个日志文件的最大尺寸。

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-handler=ACCOUNTS_TRACE:write-attribute(name="rotate-size", value="50m"), {"outcome" => "success"}
[standalone@localhost:9999 /]
```

设置要保留的备份日志的最大数目。

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="max-backup-index", value="NUMBER")
```

用文件日志处理程序的名称替换 *HANDLER*，并用要保留的日志文件的数目替换 *NUMBER*。

例 14.48. 设置要保留的备份日志的最大数目。

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-handler=ACCOUNTS_TRACE:write-attribute(name="max-backup-index", value="5")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

为 **size-rotating-file-handler** 设置 **rotate-on-boot** 选项

这个选项只用于 **size-rotating-file-handler** 文件处理程序。它的默认值是 **size-rotating-file-handler**，表示在服务器重启时不创建新的日志文件。

要修改它，请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="rotate-on-boot", value="BOOLEAN")
```

用 **size-rotating-file-handler** 日志处理程序的名称替换 *HANDLER*。如果在重启应该创建新的 **size-rotating-file-handler** 日志文件，则请用 **true** 替换 *BOOLEAN*。

例 14.49. 指定在服务器重启时创建新的 size-rotating-file-handler 日志文件。

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-handler=ACCOUNTS_TRACE:write-attribute(name="rotate-on-boot", value="true")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

删除日志处理程序

使用 **remove** 操作和下列语法。

```
/subsystem=logging/size-rotating-file-handler=HANDLER:remove
```

用文件日志处理程序的名称替换 *HANDLER*。

例 14.50. 删除日志处理程序

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-handler=ACCOUNTS_TRACE:remove
{"outcome" => "success"}
```


[提交 bug 报告](#)

14.3.7. 在 CLI 里配置 Async 日志处理程序

您可以在 CLI 里添加、删除和编辑异步（Async）日志处理程序。

配置异步日志处理程序的主要任务是：

- 添加新的 Async 日志处理程序。
- 显示异步日志处理程序的配置
- 改变日志级别
- 设置队列长度
- 设置溢出动作
- 添加子处理程序
- 删除子处理程序
- 删除异步日志处理程序

每个任务都将在下面进行描述。



重要

当在日志配置集里配置日志处理程序时，配置路径的根目录是 `/subsystem=logging/logging-profile=NAME/` 而不是 `/subsystem=logging/`。

添加新的 Async 日志处理程序。

使用 **add** 操作和下列语法。

```
/subsystem=logging/async-handler=HANDLER:add(queue-length="LENGTH")
```

用文件日志处理程序的名称替换 *HANDLER*，并用保持在队列里的日志请求的最大数目替换 *LENGTH*。

例 14.51.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:add(queue-length="10")
{"outcome" => "success"}
```

显示异步日志处理程序的配置

使用 **read-resource** 操作和下列语法。

```
/subsystem=logging/async-handler=HANDLER:read-resource
```

用文件日志处理程序的名称替换 *HANDLER*。

例 14.52.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:read-resource
{
    "outcome" => "success",
    "result" => {
        "encoding" => undefined,
        "filter" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => undefined,
        "overflow-action" => "BLOCK",
        "queue-length" => "50",
        "subhandlers" => undefined
    }
}
[standalone@localhost:9999 /]
```

改变日志级别

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/async-handler=HANDLER:write-attribute(name="level",
value="LOG_LEVEL_VALUE")
```

用日志处理程序的名称替换 *HANDLER*，并用日志级别替换 *LOG_LEVEL_VALUE*。

例 14.53.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:write-attribute(name="level", value="INFO")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

设置队列长度

请使用 **write-attribute** 命令和下列语法。

```
/subsystem=logging/async-handler=HANDLER:write-attribute(name="queue-
length", value="LENGTH")
```

用文件日志处理程序的名称替换 *HANDLER*，并用保持在队列里的日志请求的最大数目替换 *LENGTH*。

重启 JBoss EAP 6 服务器以使修改生效。

例 14.54.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:write-attribute(name="queue-length", value="150")
{
```

```

        "outcome" => "success",
        "response-headers" => {
            "operation-requires-reload" => true,
            "process-state" => "reload-required"
        }
    }
}

```

设置溢出动作

请使用 **write-attribute** 命令和下列语法。

```

/subsystem=logging/async-handler=HANDLER:write-attribute(name="overflow-action", value="ACTION")

```

用文件日志处理程序的名称替换 *HANDLER*，并用 *DISCARD* 或 *BLOCK* 替换 *ACTION*。

例 14.55.

```

[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:write-attribute(name="overflow-action", value="DISCARD")
{"outcome" => "success"}
[standalone@localhost:9999 /]

```

添加子处理程序

使用 **add-handler** 操作和下列语法。

```

/subsystem=logging/async-handler=HANDLER:add-handler(name="SUBHANDLER")

```

用文件日志处理程序的名称替换 *HANDLER*，并用将被添加为子处理程序的日志处理程序的名称替换 *SUBHANDLER*。

例 14.56.

```

[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:add-handler(name="NFS_FILE")
{"outcome" => "success"}
[standalone@localhost:9999 /]

```

删除子处理程序

使用 **remove-handler** 操作和下列语法。

```

/subsystem=logging/async-handler=HANDLER:remove-handler(name="SUBHANDLER")

```

用文件日志处理程序的名称替换 *HANDLER*，并用要删除的子处理程序的名称替换 *SUBHANDLER*。

例 14.57.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:remove-handler(name="NFS_FILE")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

删除异步日志处理程序

使用 **remove** 操作和下列语法。

```
/subsystem=logging/async-handler=HANDLER:remove
```

用文件日志处理程序的名称替换 *HANDLER*。

例 14.58.

```
[standalone@localhost:9999 /] /subsystem=logging/async-
handler=NFS_LOGS:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[提交 bug 报告](#)

14.3.8. 配置 syslog 处理程序

JBoss EAP 6 的 lognamager 现在包含一个 Syslog 处理程序。Syslog 处理程序用于发送消息到支持 **Syslog** 协议 (RFC-3164 or RFC-5424) 的远程日志服务器。它允许多个应用程序发送它们的日志消息到相同的服务器，一起进行解析。本节介绍如何通过管理 CLI 和可用的配置选项创建和配置处理程序。

- 管理 CLI 的访问权限

过程 14.1. 添加 syslog 处理程序

- 运行下列命令来添加 syslog 处理程序

```
/subsystem=logging/syslog-handler=HANDLER_NAME:add
```

过程 14.2. 配置 syslog 处理程序

- 运行下列命令来配置 syslog 处理程序属性：

```
/subsystem=logging/syslog-handler=HANDLER_NAME:write-
attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

过程 14.3. 删除 syslog 处理程序

- 运行以下命令删除现有的 Syslog 处理程序：

```
/subsystem=logging/syslog-handler=HANDLER_NAME:remove
```

表 14.6. Syslog 处理程序的配置属性

属性	描述	默认值
port	Syslog 服务器侦听的端口。	514
app-name	以 RFC5424 格式化消息时使用的 App 名称。	null
enabled	如果设置为 true，启用处理程序。如果设置为 false，处理日志消息时将忽略处理程序。	true
level	指定哪些消息将被记录的日志级别。低于这个级别的消息将被忽略。	ALL
facility	依照 RFC-5424 所 RFC-3164 定义的	user-level
server-address	Syslog 服务器的地址	localhost
hostname	发送消息的主机的名称。	null
syslog-format	按照 RFC 规格格式化日志消息	RFC5424

[提交 bug 报告](#)

14.3.9. 在 CLI 里配置自定义日志格式器

概述

除了 [第 14.1.15 节 “日志格式器语法”](#) 里指定的日志格式器语法，您可以创建自定义的日志格式器和任何日志处理程序一起使用。这个例子将为 Console 日志处理程序创建 XML 格式器以演示这一点。

必须具备的条件

- 对 JBoss EAP 6 服务器的管理 CLI 的访问权限
- 之前配置的日志处理程序。这个例子使用了 Console 日志处理程序。

过程 14.4. 配置用于日志处理程序的自定义 XML 格式器

1. 参见自定义格式器。

在这个例子里，下面的命令创建了一个名为 **XML_FORMATTER** 的自定义格式器，它使用了 `java.util.logging.XMLFormatter` 类。

```
[standalone@localhost:9999 /] /subsystem=logging/custom-formatter=XML_FORMATTER:add(class=java.util.logging.XMLFormatter, module=org.jboss.logmanager)
```

2. 为日志处理程序注册自定义格式器。

在这个例子里，之前步骤里的格式器将被添加至 **Console** 日志处理程序。

```
[standalone@localhost:9999 /] /subsystem=logging/console-  
handler=HANDLER:write-attribute(name=named-formatter,  
value=XML_FORMATTER)
```

3. 重启 JBoss EAP 6 服务器以使修改生效。

```
[standalone@localhost:9999 /] shutdown --restart=true
```

结果

自定义的 XML 格式器被添加到 **Console** 日志处理程序。控制台日志里的输出将以 XML 进行格式化，例如：

```
<record>  
  <date>2014-03-11T13:02:53</date>  
  <millis>1394539373833</millis>  
  <sequence>116</sequence>  
  <logger>org.jboss.as</logger>  
  <level>INFO</level>  
  <class>org.jboss.as.server.BootstrapListener</class>  
  <method>logAdminConsole</method>  
  <thread>282</thread>  
  <message>JBAS015951: Admin console listening on http://%s:%d</message>  
  <param>127.0.0.1</param>  
  <param>9990</param>  
</record>
```

[提交 bug 报告](#)

14.4. 重新部署日志子系统

14.4.1. 关于 Per-deployment 日志

Per-deployment 日志允许开发人员为应用程序提前进行日志配置。当部署了应用程序时，日志将按照定义好的配置进行记录。通过这个配置创建的日志文件仅包含和这个应用程序行为相关的信息。

将这种方法用于系统范围的日志时既有优点也有缺点。其优点是 **JBoss EAP** 实例的管理员不需要配置日志；缺点是 **per-deployment** 配置在启动时是只读的，所以也无法在运行时进行修改。

[提交 bug 报告](#)

14.4.2. 禁用 Per-deployment 日志

过程 14.5. 禁用 Per-deployment 日志

- 有两种禁用 **per-deployment** 日志的方法。第一个适用于所有 **JBoss EAP 6** 版本，而第二个只能用于 **JBoss EAP 6.3** 及更高版本。
 - **JBoss EAP 6** (所有版本)
添加系统属性：

```
org.jboss.as.logging.per-deployment=false
```

- **JBoss EAP 6.3 (及更高版本)**

使用 `jboss-deployment-structure.xml` 文件排除日志子系统。关于相关细节，请参考《开发指南》里的 *Exclude a Subsystem from a Deployment* 章节。

[提交 bug 报告](#)

14.5. 日志配置集

14.5.1. 关于日志配置集



重要

日志配置集只适用于 JBoss EAP 6.1.0 或以后的版本。它们不能使用管理控制台进行配置。

日志配置集 (**Logging Profile**) 是独立的日志配置，它可以分配到部署的应用程序。和普通的日志子系统一样，日志配置集可以定义处理程序、类别和 **Root Logger**，但它不能在其他配置集或主日志系统里引用配置。日志配置集的设计简便了日志系统的配置。

日志配置集允许管理员为一个或多个应用程序专门创建日志配置，而不会影响到其他的日志配置。因为每个配置集都定义在服务器配置里，所以您可以修改日志配置而无需重部署受影响的应用程序。

每个日志配置集都可以有下列配置：

- 唯一名称。这是必需的配置。
- 任意数量的日志处理程序。
- 任意数量的日志类别。
- 最多一个 **Root Logger**。

应用程序可以用 **logging-profile** 属性指定一个日志配置集来使用它的 **MANIFEST.MF** 文件。

[提交 bug 报告](#)

14.5.2. 用 CLI 创建新的日志配置集

您可以用下面的 CLI 命令创建新的日志配置集，请用自己的配置集名替换 **NAME**。

```
/subsystem=logging/logging-profile=NAME:add
```

这将创建一个可以添加处理程序、类别和 **Root Logger** 的新的空配置集。

[提交 bug 报告](#)

14.5.3. 用 CLI 配置日志配置集

日志配置集可以用和配置主日志子系统几乎相同的语法来配置日志处理程序、类别和 **Root Logger**。

配置日志子系统和日志配置集只有两个不同之处：

1. 根配置路径是 `/subsystem=logging/logging-profile=NAME`
2. 日志配置集不能包含其他日志配置集。

请参考合适的日志管理任务：

- [第 14.3.1 节 “用 CLI 配置 Root Logger”](#)
- [第 14.3.2 节 “在 CLI 里配置日志类别”](#)
- [第 14.3.3 节 “在 CLI 里配置控制台日志处理程序”](#)
- [第 14.3.4 节 “在 CLI 里配置文件日志处理程序”](#)
- [第 14.3.5 节 “在 CLI 里配置定期日志处理程序”](#)
- [第 14.3.6 节 “在 CLI 里配置 Size 日志处理程序”](#)
- [第 14.3.7 节 “在 CLI 里配置 Async 日志处理程序”](#)

例 14.59. 创建和配置日志配置集

创建日志配置集并添加一个类别和文件日志处理程序。

1. 创建配置集：

```
/subsystem=logging/logging-profile=accounts-app-profile:add
```

2. 创建文件处理程序

```
/subsystem=logging/logging-profile=accounts-app-profile/file-  
handler=ejb-trace-file:add(file={path=>"ejb-trace.log", "relative-  
to"=>"jboss.server.log.dir"})
```

```
/subsystem=logging/logging-profile=accounts-app-profile/file-  
handler=ejb-trace-file:write-attribute(name="level",  
value="DEBUG")
```

3. 创建日志类别

```
/subsystem=logging/logging-profile=accounts-app-  
profile/logger=com.company.accounts.ejbs:add(level=TRACE)
```

4. 分配文件处理程序给类别

```
/subsystem=logging/logging-profile=accounts-app-  
profile/logger=com.company.accounts.ejbs:add-handler(name="ejb-  
trace-file")
```

[提交 bug 报告](#)

14.5.4. 指定应用程序里的日志配置集

应用程序在 **MANIFEST.MF** 文件里指定要使用的日志配置集。

预备条件：

1. 您必须知道服务器上为这个应用程序设置的日志配置集的名称，您可以咨询服务器管理员要使用的配置集名称。

过程 14.6. 添加日志配置集到应用程序里

- **编辑 MANIFEST.MF**

如果您的应用程序没有 **MANIFEST.MF** 文件：用下列内容创建一个，用配置集名替换 **NAME**。

```
Manifest-Version: 1.0
Logging-Profile: NAME
```

如果您的应用程序已经有了 **MANIFEST.MF** 文件：添加下列行，用配置集名替换 **NAME**。

```
Logging-Profile: NAME
```

注意

如果您在使用 Maven 和 **maven-war-plugin**，您可以将 **MANIFEST.MF** 文件置于 **src/main/resources/META-INF/** 并添加下列配置到 **pom.xml** 文件里。

```
<plugin>
  <artifactId>maven-war-plugin</artifactId>
  <configuration>
    <archive>
      <manifestFile>src/main/resources/META-
INF/MANIFEST.MF</manifestFile>
    </archive>
  </configuration>
</plugin>
```

当应用程序被部署时，它会对日志消息使用指定日志配置集里的配置。

[提交 bug 报告](#)

14.5.5. 日志配置集配置示例

这个例子展示了日志配置集的配置以及使用它的应用程序。它也展示了 CLI 会话、生成的 XML 配置以及应用程序的 **MANIFEST.MF** 文件。

这个日志配置集示例具有下列特点：

- 名称是 **accounts-app-profile**。
- 日志类别是 **com.company.accounts.ejbs**。
- 日志级别是 **TRACE**。
- 日志处理程序是使用 **ejb-trace.log** 的文件处理程序。

例 14.60. CLI 会话

```
localhost:bin user$ ./jboss-cli.sh -c
[standalone@localhost:9999 /] /subsystem=logging/logging-
profile=accounts-app-profile:add
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-
profile=accounts-app-profile/file-handler=ejb-trace-file:add(file=
{path=>"ejb-trace.log", "relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-
profile=accounts-app-profile/file-handler=ejb-trace-file:write-
attribute(name="level", value="DEBUG")
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-
profile=accounts-app-
profile/logger=com.company.accounts.ejbs:add(level=TRACE)
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-
profile=accounts-app-profile/logger=com.company.accounts.ejbs:add-
handler(name="ejb-trace-file")
{"outcome" => "success"}

[standalone@localhost:9999 /]
```

例 14.61. XML 配置

```
<logging-profiles>
  <logging-profile name="accounts-app-profile">
    <file-handler name="ejb-trace-file">
      <level name="DEBUG"/>
      <file relative-to="jboss.server.log.dir" path="ejb-
trace.log"/>
    </file-handler>
    <logger category="com.company.accounts.ejbs">
      <level name="TRACE"/>
      <handlers>
        <handler name="ejb-trace-file"/>
      </handlers>
    </logger>
  </logging-profile>
</logging-profiles>
```

例 14.62. 应用程序的 MANIFEST.MF 文件

```
Manifest-Version: 1.0
Logging-Profile: accounts-app-profile
```

[提交 bug 报告](#)

14.6. 日志配置属性

14.6.1. Root Logger 属性

表 14.7. Root Logger 属性

属性	数据类型	描述
level	字符串	root logger 记录的最大级别的日志消息。
handlers	String[]	Root Logger 使用的处理程序列表。
filter-spec	字符串	定义过滤器的表达式，下面的表达式定义了一个过滤器，它排除不匹配某个模式的条目，如： <code>not(match("JBAS.*"))</code> 。



注意

为 root logger 指定的 **filter-spec** 不是从其他处理程序继承的。您必须为每个处理程序指定一个 **filter-spec**。

[提交 bug 报告](#)

14.6.2. 日志类别属性

表 14.8. 日志类别属性

属性	数据类型	描述
level	字符串	日志类别记录的最大级别的日志消息。
handlers	String[]	Root Logger 使用的处理程序列表。
use-parent-handlers	布尔值 (Boolean)	如果为 true，除了其他分配的处理程序外，这个列表将使用 Root Logger 日志处理程序。
category	字符串	哪个日志类别里的消息将被记录。
filter-spec	字符串	定义过滤器的表达式。下面的表达式定义了一个不匹配某个模式的过滤器： <code>not(match("JBAS.*"))</code> 。

[提交 bug 报告](#)

14.6.3. Console Log Handler 的属性

表 14.9. Console Log Handler 的属性

属性	数据类型	描述
level	字符串	日志处理程序记录的最大级别的日志消息。
encoding	字符串	输出所使用的编码模式。
formatter	字符串	这个日志处理程序使用的日志格式器。
target	字符串	日志处理程序的输出所去往的系统输出流，这可以是系统错误流的 <code>System.err</code> 或标准输出流的 <code>System.out</code> 。
autoflush	布尔值 (Boolean)	如果为 <code>true</code> ，日志消息在接收后将立即被送往处理程序目标。
name	字符串	这个日志处理程序的唯一标识。
enabled	布尔值 (Boolean)	如果设置为 <code>true</code> ，启用处理程序。如果设置为 <code>false</code> ，处理日志消息时将忽略处理程序。
filter-spec	字符串	定义过滤器的表达式。下面的表达式定义了一个不匹配某个模式的过滤器： <code>not(match("JBAS.*"))</code> 。

[提交 bug 报告](#)

14.6.4. 文件处理程序属性

表 14.10. 文件处理程序属性

属性	数据类型	描述
level	字符串	日志处理程序记录的最大级别的日志消息。
encoding	字符串	输出所使用的编码模式。
formatter	字符串	这个日志处理程序使用的日志格式器。
append	布尔值 (Boolean)	如果为 <code>true</code> ，这个处理程序写入的所有日志消息将附加在文件（如果已经存在）后面。如果为 <code>false</code> ，每次应用程序启动时都会创建一个新的文件。对 append 的修改要求服务器重启以使其生效。
autoflush	布尔值 (Boolean)	如果为 <code>true</code> ，日志消息在接收后将立即被送往处理程序。对 autoflush 的修改要求服务器重启以使其生效。
name	字符串	这个日志处理程序的唯一标识。
file	对象	这个对象代表了日志处理程序输出写入的文件。它有两个配置属性： relative-to 和 path 。

属性	数据类型	描述
relative-to	字符串	这是文件对象的属性，也是日志文件写入的目录。在这里可以指定 JBoss EAP 6 的文件路径变量。 <code>jboss.server.log.dir</code> 变量指向服务器的 <code>log/</code> 。
path	字符串	这是文件对象的属性，也是日志消息写入的文件。它是附加在 relative-to 属性上以确定完整路径的相对路径名。
enabled	布尔值 (Boolean)	如果设置为 true ，启用处理程序。如果设置为 false ，处理日志消息时将忽略处理程序。
filter-spec	字符串	定义过滤器的表达式。下面的表达式定义了一个不匹配某个模式的过滤器： <code>not(match("JBAS.*"))</code> 。

[提交 bug 报告](#)

14.6.5. 定期日志处理程序属性

表 14.11. 定期日志处理程序属性

属性	数据类型	描述
append	布尔值 (Boolean)	如果为 true ，这个处理程序写入的所有日志消息将附加在文件（如果已经存在）后面。如果为 false ，每次应用程序启动时都会创建一个新的文件。对 append 属性的修改要求服务器重启以使其生效。
autoflush	布尔值 (Boolean)	如果为 true ，日志消息在接收后将立即被送往处理程序。对 autoflush 的修改要求服务器重启以使其生效。
encoding	字符串	输出所使用的编码模式。
formatter	字符串	这个日志处理程序使用的日志格式器。
level	字符串	日志处理程序记录的最大级别的日志消息。
name	字符串	这个日志处理程序的唯一标识。
file	对象	这个对象代表了日志处理程序输出写入的文件。它有两个配置属性： relative-to 和 path 。
relative-to	字符串	这是文件对象的属性，也是日志文件写入的目录。在这里可以指定 JBoss EAP 6 的文件路径变量。 <code>jboss.server.log.dir</code> 变量指向服务器的 <code>log/</code> 。
path	字符串	这是文件对象的属性，也是日志消息写入的文件。它是附加在 relative-to 属性上以确定完整路径的相对路径名。

属性	数据类型	描述
suffix	字符串	<p>这个字符串附加在轮换日志文件的名称后面，用于确定轮换的频率。后缀的格式是 (.) 加上 <code>java.text.SimpleDateFormat</code> 类解析的日期字符串。日志文件根据后缀定义的最小时间单元进行轮换。例如，后缀 <code>.yyyy-MM-dd</code> 将每日进行轮换。</p> <p>请参考 http://docs.oracle.com/javase/6/docs/api/index.html?java/text/SimpleDateFormat.html</p>
enabled	布尔值 (Boolean)	如果设置为 true ，启用处理程序。如果设置为 false ，处理日志消息时将忽略处理程序。
filter-spec	字符串	定义过滤器的表达式。下面的表达式定义了一个不匹配某个模式的过滤器： <code>not(match("JBAS.*"))</code> 。

[提交 bug 报告](#)

14.6.6. Size 日志处理程序属性

表 14.12. Size 日志处理程序属性

属性	数据类型	描述
append	布尔值 (Boolean)	如果为 true ，这个处理程序写入的所有日志消息将附加在文件（如果已经存在）后面。如果为 false ，每次应用程序启动时都会创建一个新的文件。对 append 属性的修改要求服务器重启以使其生效。
autoflush	布尔值 (Boolean)	如果为 true ，日志消息在接收后将立即被送往处理程序。对 append 属性的修改要求服务器重启以使其生效。
encoding	字符串	输出所使用的编码模式。
formatter	字符串	这个日志处理程序使用的日志格式器。
level	字符串	日志处理程序记录的最大级别的日志消息。
name	字符串	这个日志处理程序的唯一标识。
文件	对象	这个对象代表了日志处理程序输出写入的文件。它有两个配置属性： relative-to 和 path 。
relative-to	字符串	这是文件对象的属性，也是日志文件写入的目录。在这里可以指定 JBoss EAP 6 的文件路径变量。 <code>jboss.server.log.dir</code> 变量指向服务器的 log/ 。
path	字符串	这是文件对象的属性，也是日志消息写入的文件。它是附加在 relative-to 属性上以确定完整路径的相对路径名。

属性	数据类型	描述
rotate-size	整数	在日志文件轮换前可到达的最大尺寸。数字后面的单个字符表示单位： b 表示字节、 k 表示千字节、 m 表示兆字节、 g 表示千兆字节、 50m 表示 50 兆字节。
max-backup-index	整数	设置要保留的轮换日志的最大数目。到达这个数目时，最旧的日志将被重用。
enabled	布尔值 (Boolean)	如果设置为 true ，启用处理程序。如果设置为 false ，处理日志消息时将忽略处理程序。
filter-spec	字符串	定义过滤器的表达式。下面的表达式定义了一个不匹配某个模式的过滤器： not(match("JBAS.*")) 。
rotate-on-boot	布尔值 (Boolean)	如果为 true ，服务器重启时将创建新的日志文件。默认值是 false 。

[提交 bug 报告](#)

14.6.7. 异步日志处理程序属性

表 14.13. 异步日志处理程序属性

属性	数据类型	描述
level	字符串	日志处理程序记录的最大级别的日志消息。
name	字符串	这个日志处理程序的唯一标识。
queue-length	整数	在等待子处理程序响应时这个处理程序保留的日志消息的最大数目。
overflow-action	字符串	当超过队列长度时，这个处理程序如何响应。它可以设置为 BLOCK 或 DISCARD 。 BLOCK 让日志应用程序等待，直至队列里出现可用的空间。这和非异步日志处理程序有着相同的行为。 DISCARD 则允许日志应用程序继续，但会删除日志消息。
subhandlers	String[]	这个异步处理程序传入日志消息的处理程序的列表。
enabled	布尔值 (Boolean)	如果设置为 true ，启用处理程序。如果设置为 false ，处理日志消息时将忽略处理程序。
filter-spec	字符串	定义过滤器的表达式。下面的表达式定义了一个不匹配某个模式的过滤器： not(match("JBAS.*")) 。

[提交 bug 报告](#)

14.7. 用于日志的 XML 配置示例

14.7.1. 用于 Root Logger 的 XML 配置示例

```
<root-logger>
  <level name="INFO"/>
  <handlers>
    <handler name="CONSOLE"/>
    <handler name="FILE"/>
  </handlers>
</root-logger>
```

[提交 bug 报告](#)

14.7.2. 用于日志类别的 XML 配置示例

```
<logger category="com.company.accounts.rec">
  <handlers>
    <handler name="accounts-rec"/>
  </handlers>
</logger>
```

[提交 bug 报告](#)

14.7.3. 用于 Console Log Handler 的 XML 配置示例

```
<console-handler name="CONSOLE">
  <level name="INFO"/>
  <formatter>
    <pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/>
  </formatter>
</console-handler>
```

[提交 bug 报告](#)

14.7.4. 用于 File 日志处理程序的 XML 配置示例

```
<file-handler name="accounts-rec-trail" autoflush="true">
  <level name="INFO"/>
  <file relative-to="jboss.server.log.dir" path="accounts-rec-
trail.log"/>
  <append value="true"/>
</file-handler>
```

[提交 bug 报告](#)

14.7.5. 用于 Periodic 日志处理程序的 XML 配置示例

```
<periodic-rotating-file-handler name="FILE">
  <formatter>
    <pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t)
%s%E%n"/>
  </formatter>
</periodic-rotating-file-handler>
```



```
</formatter>
<file relative-to="jboss.server.log.dir" path="server.log"/>
<suffix value=".yyyy-MM-dd"/>
<append value="true"/>
</periodic-rotating-file-handler>
```

[提交 bug 报告](#)

14.7.6. 用于 **Size** 日志处理程序的 **XML** 配置示例

```
<size-rotating-file-handler name="accounts_debug" autoflush="false">
  <level name="DEBUG"/>
  <file relative-to="jboss.server.log.dir" path="accounts-debug.log"/>
  <rotate-size value="500k"/>
  <max-backup-index value="5"/>
  <append value="true"/>
</size-rotating-file-handler>
```

[提交 bug 报告](#)

14.7.7. 用于 **Async** 日志处理程序的 **XML** 配置示例

```
<async-handler name="Async_NFS_handlers">
  <level name="INFO"/>
  <queue-length value="512"/>
  <overflow-action value="block"/>
  <subhandlers>
    <handler name="FILE"/>
    <handler name="accounts-record"/>
  </subhandlers>
</async-handler>
```

[提交 bug 报告](#)

第 15 章 INFINISPAN

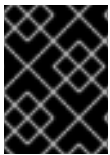
15.1. 关于 INFINISPAN

Infinispan 是一个 Java 数据网格平台。它为管理缓存数据提供了一个兼容 [JSR-107](#) 的缓存接口。

JBoss EAP 6 里使用了下列 Infinispan 缓存容器：

- 用于 Web 会话群集的 **web**
- 用于 Stateful Session Bean 群集的 **ejb**
- 用于实体缓存的 **hibernate**
- 用于单点登录缓存的 **singleton**

每个缓存容器都定义了一个 "repl" 和一个 "dist" 缓存。用户应用程序不应该直接使用这些缓存。



重要

用户可以添加更多的缓存容器和缓存，并通过 JNDI 引用它们。然而，JBoss EAP 6 并不支持这样做。

关于 Infinispan 的功能和配置选项的更多信息，请参考 [Infinispan 文档](#)。

[提交 bug 报告](#)

15.2. 群集模式

在 JBoss EAP 6 里您可以用 Infinispan 以两种方式配置群集。正确的方法取决于您的要求。每种模式下可用性、一致性、可靠性和可扩充性之间都需要取舍。在选择群集模式之前，您必须根据您的网络确定最重要的方面，并平衡其他要求。

复制模式

复制模式 (Replicated Mode) 自动检测并添加群集上的新实例。对这些实例的修改将复制到群集里的所有节点上。小型群集里复制模式通常是最佳选择，这是因为在网络上复制的信息量较少。您可以配置 Infinispan 来使用 UDP 多点传送，它可以在某种程度上将网络堵塞减缓。

分布模式

分布模式 (Distribution Mode) 允许 Infinispan 线性地扩充群集。分布模式使用一致的哈希算法来确定新节点应该放置在哪个群集里。保留的信息的拷贝数量是可以配置的。保留的信息拷贝和数据的持久性及性能之间需要进行取舍：保留越多的拷贝，性能越受到影响，但在服务器发生鼓掌时越不可能丢失数据。通过定位入口而无需多点传送或存储元数据，哈希算法也可以减少网络流量。

同步和异步复制

复制可以同步或异步模式执行，所选的模式取决于您的要求及应用程序。如使用同步复制，处理用户请求的线程将阻塞至复制成功。只有在复制成功时响应才会发送给客户，然后释放线程。同步复制将对网络流量有影响，因为它要求得到群集里每个节点的响应。然而，它的优势是可以确保所有的修改都已应用于群集的所有节点。

异步复制是在后台执行的。Infinispan 实现是一个复制队列，后台线程使用它来执行复制。复制通过时间或者队列大小来触发。复制队列可提高性能，因为在群集节点间没有进行对话。异步复制的缺点是不是很准确。失败的复制尝试会写入日志，而不是实时通知。

[提交 bug 报告](#)

15.3. 缓存容器

缓存容器

缓存容器是子系统使用的缓存的资料库。Infinispan 的默认缓存容器是在 XML 配置文件（standalone-ha.xml、standalone-full-ha.xml、domain.xml）里定义的。其中一个会定义为默认缓存，它将用于群集。

例 15.1. standalone-ha.xml 配置文件里的缓存容器定义：

```
<subsystem xmlns="urn:jboss:domain:infinispan:1.5">
  <cache-container name="singleton" aliases="cluster ha-
partition" default-cache="default">
    <transport lock-timeout="60000"/>
    <replicated-cache name="default" mode="SYNC"
batching="true">
      <locking isolation="REPEATABLE_READ"/>
    </replicated-cache>
  </cache-container>
  <cache-container name="web" aliases="standard-session-cache"
default-cache="repl" module="org.jboss.as.clustering.web.infinispan">
    <transport lock-timeout="60000"/>
    <replicated-cache name="repl" mode="ASYNC" batching="true">
      <file-store/>
    </replicated-cache>
    <replicated-cache name="sso" mode="SYNC" batching="true"/>
    <distributed-cache name="dist" l1-lifespan="0"
mode="ASYNC" batching="true">
      <file-store/>
    </distributed-cache>
  </cache-container>
```

请注意每个缓存容器里定义的默认缓存。在这个例子的 **web** 缓存容器里，**repl** 被定义为默认缓存。因此在群集 **Web** 会话时 **repl** 将被使用。

您可以用管理控制器或 CLI 命令来配置缓存容器及缓存属性。但我们不建议修改缓存容器或缓存的名称。

配置缓存容器

你可以用管理 CLI 或管理控制台来配置 Infinispan 的缓存容器。

过程 15.1. 在管理控制台里配置 Infinispan 的缓存容器

1. 从屏幕顶部选择 **Configuration** 标签页。
2. 如果服务器运行于域模式，请从左上角的下拉菜单里选择 **ha** 或 **full-ha**。
3. 展开 **Subsystems** 菜单，然后展开 **Infinispan** 菜单。选择 **Cache Containers**。
4. 从 **Cache Containers** 表里选择缓存容器。
5. 添加、删除或设置默认的缓存容器

- a. 要创建新的缓存容器，点击 **Cache Containers** 表里的 **Add** 按钮。
 - b. 要删除缓存容器，从 **Cache Containers** 表里选择缓存容器，点击 **Remove**，然后点击 **OK** 确认。
 - c. 要设置默认的缓存容器，点击 **Set Default**，在下拉列表里输入缓存容器的名称，点击 **Save** 确认。
6. 要添加或更新缓存容器的属性，请在 **Cache Containers** 表里选择缓存容器。在屏幕的 **Details** 区域里的 **Attributes**、**Transport** 和 **Aliases** 标签页里选择缓存容器，然后点击 **Edit**。关于 **Attributes**、**Transport** 和 **Aliases** 标签页里内容的帮助，请点击 **Need Help?**。

过程 15.2. 在管理 CLI 里配置 Infinispan 的缓存容器

1. 要获取可配置属性的列表，请输入下列 CLI 命令：

```
/profile=profile name/subsystem=infinispan/cache-
container=container name:read-resource
```

2. 您可以使用管理 CLI 来添加、删除和更新缓存容器。在输入和缓存容器相关的任何命令之前，请确保您在 CLI 命令里使用了正确的配置集。

- a. **添加缓存容器**

要添加缓存容器，请以下列命令为基础：

```
/profile=profile-name/subsystem=infinispan/cache-
container="cache container name":add
```

- b. **删除缓存容器**

要删除缓存容器，请以下列命令为基础：

```
/profile=profile-name/subsystem=infinispan/cache-
container="cache container name":remove
```

- c. **更新缓存容器属性**

使用 **write-attribute** 操作来编写新的属性的值。您可以使用 **tab completion** 功能来帮助输入并提示所有可用的值。下面的例子将 **statistics-enabled** 更新为 **true**。

```
/profile=profile name/subsystem=infinispan/cache-
container=cache container name:write-attribute(name=statistics-
enabled,value=true)
```

[提交 bug 报告](#)

15.4. 缓存库

缓存库 (Cache Store) 是缓存里数据的一个外部存储。和 JBoss EAP 6 最为相关的外部数据存储库类型是基于文件的、基于 JDBC 的以及远程 Infinispan/JDG 库。

对于基于文件的缓存库，群集里的每个节点通常都有自己的文件系统及基于文件的缓存库。我们不建议将基于文件的缓存库置于共享文件系统（NFS 等）里，因为它们无法实现正确的文件锁，可能会导致数据损坏。

对于基于 JDBC 的缓存库，您可以用单个 SQL 数据库充当所有群集节点的缓存库。然而，您必须将缓存库配置为共享的（设置基于 JDBC 的缓存库上的 **shared** 属性为 **true**）。如果没有定义缓存库为共享的，就可能发生数据库死锁以及其他影响性能的问题。

[提交 bug 报告](#)

15.5. 关于 INFINISPAN 统计

您可以启用关于 Infinispan 缓存和缓存容器对象的运行时统计以进行监控。处于性能考虑，统计采集默认是禁用的。



警告

启用 Infinispan 统计对 Infinispan 子系统的性能有负面影响。请在有需要时才启用统计。

您可以为每个缓存容器、缓存或两者都启用统计采集。每个缓存的统计选项都会覆盖缓存容器的选项。启用或禁用缓存容器的统计采集将导致该容器里所有缓存都继承它的设置，除非显性地指定自己的设置。如果只有缓存容器启用了统计，容器里的缓存都会进行统计信息采集。

[提交 bug 报告](#)

15.6. 启用 INFINISPAN 统计信息的采集

您可以在启动配置文件（**standalone.xml**、**standalone-ha.xml** 或 **domain.xml**）或管理 CLI 里启用统计采集。

[提交 bug 报告](#)

15.6.1. 在启动配置文件里启用 Infinispan 统计信息的采集

过程 15.3. 在启动配置文件里启用 Infinispan 统计

- 在 Infinispan 子系统的 **cache-container** 或 **cache XML** 标签下添加 **statistics-enabled=VALUE** 属性。

例 15.2. 为 cache 启用统计采集

```
<replicated-cache name="sso" mode="SYNC" batching="true" statistics-enabled="true"/>
```

例 15.3. 为 cache-container 启用统计采集

```
<cache-container name="singleton" aliases="cluster ha-partition"
default-cache="default" statistics-enabled="true">
```

[提交 bug 报告](#)

15.6.2. 从管理 CLI 启用 Infinispan 统计信息的采集

过程 15.4. 从管理 CLI 启用 Infinispan 统计信息的采集

在这个过程中：

- **CACHE_CONTAINER** 是首选的 **cache-container**（例如，**web**）。
- **CACHE_TYPE** 是首选的缓存类型（例如，**distributed-cache**）。
- **CACHE** 是缓存名称（例如，**dist**）。

1. 输入下列命令：

```
/subsystem=infinispan/cache-
container=CACHE_CONTAINER/CACHE_TYPE=CACHE:write-
attribute(name=statistics-enabled,value=true)
```

2. 输入下列命令来重载服务器：

```
:reload
```



注意

要取消属性定义，请输入下列命令：

```
/subsystem=infinispan/cache-
container=CACHE_CONTAINER/CACHE_TYPE=CACHE:undefine-
attribute(name=statistics-enabled)
```

[提交 bug 报告](#)

15.6.3. 检验是否已启用 Infinispan 统计信息的采集

过程 15.5. 检验是否已启用 Infinispan 统计信息的采集

根据您的确认是在 **cache** 还是 **cache-container** 上启用统计采集，请使用下列管理 CLI 命令。

- ○ 对于 **cache**

```
/subsystem=infinispan/cache-  
container=CACHE_CONTAINER/CACHE_TYPE=CACHE:read-  
attribute(name=statistics-enabled)
```

◦ 对于 **cache-container**

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:read-  
attribute(name=statistics-enabled)
```

[提交 bug 报告](#)

15.7. JGROUPS

15.7.1. 关于 JGroups

JGroups 是一个消息工具包，它允许开发人员创建可以运行在可能有问题的系统里的可靠的消息应用程序。JGroups 可以用来创建群集，其中节点可以发送消息到彼此。

JGroups 子系统为群集里服务器如何与彼此通讯提供了所有的通讯机制。EAP 预配置了两个 JGroups 栈。

- **udp** - 群集里的节点使用 UDP (User Datagram Protocol) 多点传送来进行彼此通讯。UDP 通常比 TCP 快但不如它可靠。
- **tcp** - 群集里的节点使用 TCP (Transmission Control Protocol) 来进行彼此通讯。TCP 比 UDP 慢但递送数据到目的地更为可靠。

您可以使用预配置的栈，或者您可以定义自己的栈来满足系统的需要。

[提交 bug 报告](#)

第 16 章 JVM

16.1. 关于 JVM

16.1.1. 关于 JVM 设置

对于受管域和独立服务器实例，Java 虚拟机（JVM）设置是不同的。在受管域里，JVM 设置是在 **host.xml** 和 **domain.xml** 配置文件里声明的，由负责启动和停止服务器进程的域控制器组件确定。而在独立服务器实例里，服务器的启动过程可以传入命令行参数。这可以通过命令行或管理控制台的 **System Properties** 屏幕里进行声明。

受管域

受管域的一个重要特征是可以定义多级别的 JVM 设置。您可以在主机、服务器组、服务器实例级别配置定义 JVM 设置。您可以用更专用的子元素覆盖父配置，这允许声明专有的服务器配置而无需在组或主机级别进行排除。这也允许其他级别继承父配置，直至在运行时传入或在配置文件里声明设置。

例 16.1. 域配置文件里的 JVM 设置

下面的例子展示了 **domain.xml** 配置文件里的服务器组的 JVM 声明。

```
<server-groups>
  <server-group name="main-server-group" profile="default">
    <jvm name="default">
      <heap size="64m" max-size="512m"/>
    </jvm>
    <socket-binding-group ref="standard-sockets"/>
  </server-group>
  <server-group name="other-server-group" profile="default">
    <jvm name="default">
      <heap size="64m" max-size="512m"/>
    </jvm>
    <socket-binding-group ref="standard-sockets"/>
  </server-group>
</server-groups>
```

在这个例子里，名为 **main-server-group** 的组声明了大小为 **64MB** 的堆，其最大堆大小为 **512MB**。属于这个组的任何服务器都将继承这些设置。你可以为整个组、为某个主机、或者单个服务器修改这些设置。

例 16.2. 主机配置文件里的域设置

下面的例子展示了 **host.xml** 配置文件里的服务器组的 JVM 声明。

```
<servers>
  <server name="server-one" group="main-server-group" auto-start="true">
    <jvm name="default"/>
  </server>
  <server name="server-two" group="main-server-group" auto-start="true">
    <jvm name="default">
      <heap size="64m" max-size="256m"/>
    </jvm>
  </server>
</servers>
```



```

        <socket-bindings port-offset="150"/>
    </server>
    <server name="server-three" group="other-server-group" auto-
start="false">
        <socket-bindings port-offset="250"/>
    </server>
</servers>

```

在这个例子里，名为 **server-two** 的服务器属于 **main-server-group** 服务器组，它继承了 **default JVM** 组的设置。在前面的例子里，**main-server-group** 的主要堆大小是 512MB。通过声明更小的最大堆尺寸为 256MB，**server-two** 可以覆盖 **domain.xml** 设置来按需要调整性能。

运行时的独立服务器设置

独立服务器实例的 JVM 设置可以在启动服务器前通过设置 **JAVA_OPTS** 环境变量来声明。下面是在 Linux 命令行里设置 **JAVA_OPTS** 环境变量的例子：

```
[user@host bin]$ export JAVA_OPTS="-Xmx1024M"
```

相同的设置可以用于 Microsoft Windows 环境，如：

```
C:\> set JAVA_OPTS="-Xmx1024M"
```

或者，可以将 JVM 设置添加到 **EAP_HOME/bin** 目录下的 **standalone.conf** 文件里，它包含了传入 JVM 的选项示例。



警告

设置 **JAVA_OPTS** 环境变量将重新定义 **JAVA_OPTS** 的默认值。这可以打断或终止 EAP 的启动。

[提交 bug 报告](#)

16.1.2. 在管理控制台里显示 JVM 状态

必须具备的条件

- [第 2.1.2 节 “将 JBoss EAP 6 作为独立服务器启动”](#)
- [第 2.1.3 节 “将 JBoss EAP 6 作为受管域启动”](#)
- [第 3.4.2 节 “登录到管理控制台”](#)

独立服务器或受管域的 Java 虚拟机 (Java Virtual Machine, JVM) 状态都可以在管理控制台里显示。控制台显示了服务器的堆、线程的使用情况。虽然统计数据并非实时显示的，但您可以刷新控制台来显示 JVM 资源的最新状况。

JVM 状态包含下列数值。

表 16.1. JVM 状态属性

类型	描述
Max	可以用于内存管理的最大内存数量。最大可用的内存数用浅灰色条显示。
Used	已使用的内存数量，以深灰色的条显示。
Committed	JVM 可使用的内存数量，以深灰色条显示。
Init	JVM 从操作系统请求的用于内存管理的初始内存数量，以深灰色条显示。

过程 16.1. 在管理控制台里显示 JVM 状态

1.

显示独立服务器实例的 JVM 状态

从屏幕顶部选择 **Runtime** 标签页。展开 **Status** 菜单，然后展开 **Platform** 菜单。选择 **JVM**。

显示受管域的 JVM 状态

从屏幕顶部选择 **Runtime** 标签页。展开 **Server Status** 菜单，然后展开 **Platform** 菜单。选择 **JVM**。
2. 受管域可以提供服务器组里的所有服务器实例的可视性，但它只允许您在服务器菜单里一次查看一个服务器。要查看服务器组里其他服务器的状态，请点击屏幕左边的 **Change Server** 以选择组里显示的主机和服务器。选择所需的服务器或主机，JVM 细节将相应改变。点击 **Close** 完成。

结果

显示服务器实例的 JVM 设置的状态。

[提交 bug 报告](#)

16.1.3. 配置 JVM

<jvm></jvm> 标签支持 <jvm-options></jvm-options>，它通过 <option value=""/> 标签来添加 JVM 配置参数。

例如

```
<jvm name="default"> <heap size="1303m" max-size="1303m"/> <permgen max-size="256m"/>
<jvm-options> <option value="-XX:+UseCompressedOops"/> </jvm-options> </jvm>
```

用 CLI 配置 JVM

要用 CLI 配置 JVM，请使用以下语法：

```
# cd /server-group=main-server-group/jvm=default

# :add-jvm-option(jvm-option="-XX:+UseCompressedOops")
{
  "outcome" => "success",
  "result" => undefined,
```

```

    "server-groups" => undefined
  }

# :read-resource

# Expected Result:

[domain@localhost:9999 jvm=default] :read-resource
{
  "outcome" => "success",
  "result" => {
    "agent-lib" => undefined,
    "agent-path" => undefined,
    "env-classpath-ignored" => undefined,
    "environment-variables" => undefined,
    "heap-size" => "1303m",
    "java-agent" => undefined,
    "java-home" => undefined,
    "jvm-options" => ["-XX:+UseCompressedOops"],
    "max-heap-size" => "1303m",
    "max-permgen-size" => "256m",
    "permgen-size" => undefined,
    "stack-size" => undefined,
    "type" => undefined
  }
}

```

删除 jvm-options 条目

要删除 jvm-options 条目，请使用下列语法：

```

# cd /server-group=main-server-group/jvm=default

# :remove-jvm-option(jvm-option="-XX:+UseCompressedOops")

# Expected Result:

[domain@localhost:9999 jvm=default] :remove-jvm-option(jvm-option="-
XX:+UseCompressedOops")
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => undefined
}

```

[提交 bug 报告](#)

第 17 章 WEB 子系统

17.1. 配置 WEB 子系统

您可以用基于 Web 的管理控制台或命令行管理 CLI 配置 Web 子系统的大多数方面。我们会按管理控制台里出现的顺序逐一解释每个设置，同时也提供对管理 CLI 命令的解释。

用管理控制台查看 Web 子系统

要使用基于 Web 的管理控制台配置 Web 子系统，请点击屏幕顶部的 **Configuration** 标签页。展开 **Subsystems** 菜单，然后展开 **Web** 菜单。屏幕会显示 Web 子系统的每个可配置部分。



注意

对于受管域，只有您的配置集是 **ha** 或 **full-ha** 时，或者如果您用 **standalone-ha** 或 **standalone-full-ha** 配置集启动独立服务器时，**mod_cluster** 组件才可用；**mod_cluster** 配置的详情请参考 [第 19.5.2 节“配置 mod_cluster 子系统”](#)。

配置 JSP 容器、HTTP 连接器和虚拟 HTTP 服务器

要配置 JSP 容器、HTTP 连接器和虚拟 HTTP 服务器，请点击 **Servlet/HTTP** 菜单条目。点击 **Edit** 按钮修改任何值。点击 **Advanced** 按钮查看高级选项。下面解释了这些选项。HTTP 连接器和虚拟服务器的选项显示在单独的表里。

表 17.1. Servlet/HTTP 配置选项

选项	描述	CLI 命令
Instance ID	这个标识符用来在负载均衡场景里启用会话关联性 (Session Affinity)。这个标识符必须在群里所有 JBoss EAP 服务器里是唯一的，它将附加到生成的会话标识符后面。这允许前端代理将专有会话转发到相同的 JBoss EAP 实例里。默认情况下不会设置实例 ID。	<pre>/profile=full-ha/subsystem=web:write-attribute(name=instance-id,value=worker1)</pre>
Disabled?	如果为 true ，禁用 JSP 容器。默认是 false 。如果您没有使用任何 JSP，这会很有用。	<pre>/profile=full-ha/subsystem=web/configuration=jsp-configuration/:write-attribute(name=disabled,value=false)</pre>

选项	描述	CLI 命令
Development?	如果为 true ，则启用 Development Mode ，这会产生更多冗余的调试信息。默认是 false 。	<pre>/profile=full-ha/subsystem=web/configuration=:write - attribute(name=development,value=false)</pre>
Keep Generated?	如果这个选项是隐藏的，点击 Advanced 查看这个选项。如果为 true 则保持生成的 Servlets 。它默认是 true 。	<pre>/profile=full-ha/subsystem=web/configuration=:write - attribute(name=keep-generated,value=true)</pre>
Check Interval?	如果这个选项是隐藏的，点击 Advanced 查看这个选项。这个值以秒为单位，它确定用后台进程检查 JSP 更新的频率。默认值是 0 。	<pre>/profile=full-ha/subsystem=web/configuration=:write - attribute(name=check-interval,value=0)</pre>
Display Source?	如果这个选项是隐藏的，点击 Advanced 查看这个选项。如果为 true ，当 runtime 错误发生时显示 JSP 源码片段。默认值是 true 。	<pre>/profile=full-ha/subsystem=web/configuration=:write - attribute(name=display-source-fragment,value=true)</pre>

对于负载均衡和 HA 群集，AJP 和 HTTP 连接器使用 **mod_cluster**、**mod_jk**、**mod_proxy**、**ISAPI** 和 **NSAPI**。要配置连接器，请选择 **Connectors** 标签并点击 **Add**。要删除连接器，请选中这个条目并点击 **Remove**。要编辑连接器，选中后点击 **Edit**。

如下列命令所示，当您用管理 CLI 创建新的连接器时，可以立即设置所需选项。

例 17.1. 创建新的连接器

```
/profile=full-ha/subsystem=web/connector=ajp/:add(socket-binding=ajp,scheme=http,protocol=AJP/1.3,secure=false,name=ajp,max-post-size=2097152,enabled=true,enable-lookups=false,redirect-port=8433,max-
```

save-post-size=4096)

表 17.2. 连接器选项

选项	描述	CLI 命令
Name	连接器的唯一名称，用于显示目的。	<pre>/profile=full- ha/subsystem=web/con nector=ajp/:read- attribute(name=name)</pre>
Socket Binding	连接器要绑定的命名套接字绑定。套接字绑定是套接字名称和网络端口之间的映射。套接字绑定是对每个独立服务器配置的，对于受管域则通过套接字绑定组进行配置。套接字绑定组应用于服务器组。	<pre>/profile=full- ha/subsystem=web/con nector=ajp/:write- attribute(name=socket-binding,value=ajp)</pre>
Scheme	Web 连接器 scheme，如 HTTP 或 HTTPS。	<pre>/profile=full- ha/subsystem=web/con nector=ajp/:write- attribute(name=scheme,value=http)</pre>
Protocol	要使用的 Web 连接器协议，如 AJP 或 HTTP。	<pre>/profile=full- ha/subsystem=web/con nector=ajp/:write- attribute(name=protocol,value=AJP/1.3)</pre>
Enabled	是否启用这个 Web 连接器。	<pre>/profile=full- ha/subsystem=web/con nector=ajp/:write- attribute(name=enabled,value=true)</pre>
Redirect Port	用来指定在重定向时使用的端口号。常用的重定向端口是安全端口 (https) 或 AJP 连接器。	<pre>/profile=full- ha/subsystem=web/con nector=http:write- attribute(name=redirect-port,value=8443)</pre>

选项	描述	CLI 命令
Redirect Binding	重定向绑定 (Redirect binding) 的行为和重定向端口类似，除了它要求在其“值”里指定套接字绑定名称而不是端口号。 redirect-binding 提供更高的配置灵活性，因为它允许使用预定义的特定套接字绑定端口 (HTTPS、AJP 等)。它导致和 redirect-port 选项相同的结果。	<pre>/profile=full-ha/subsystem=web/connector=http:write-attribute(name=redirect-binding,value=https)</pre>

要配置虚拟服务器，请点击 **Virtual Servers** 标签页。请用 **Add** 按钮来添加新的虚拟服务器。要编辑或删除虚拟服务器，选中相关条目并点击 **Edit** 或 **Remove** 按钮。

如下列命令所示，当您用管理 CLI 添加新的虚拟服务器时，可以立即设置所需选项。

例 17.2. 添加新的虚拟服务器

```
/profile=full-ha/subsystem=web/virtual-server=default-host/:add(enable-welcome-root=true,default-web-module=ROOT.war,alias=["localhost","example.com"],name=default-host)
```

表 17.3. 虚拟服务器选项

选项	描述	CLI 命令
Name	虚拟服务器的唯一名称，用于显示目的。	<pre>/profile=full-ha/subsystem=web/virtual-server=default-host/:read-attribute(name=name)</pre>
Alias	应该匹配这个虚拟服务器的主机名的列表。在管理控制台里，请为每一行使用一个主机名。	<pre>/profile=full-ha/subsystem=web/virtual-server=default-host/:write-attribute(name=alias,value=["localhost","example.com"])</pre>

选项	描述	CLI 命令
Default Module	模块里的 web 应用程序应该部署在虚拟服务器的根节点，且 HTTP 请求里没有给出目录时将显示。	<pre>/profile=full-ha/subsystem=web/virtual-server=default-host/:write-attribute(name=default-web-module,value=ROOT.war)</pre>

[提交 bug 报告](#)

17.2. 替换默认的 WELCOME WEB 应用程序

JBoss EAP 6 包含了一个 Welcome 应用程序，它在您访问服务器端口 8080 时显示。使用下列步骤，您可以用自己的 Web 应用程序替换这个程序。

过程 17.1. 用自己的 Web 应用程序替换默认的 Welcome 应用程序。

- 禁用 Welcome 应用程序。**
用管理 CLI 脚本 `EAP_HOME/bin/jboss-cli.sh` 运行下列命令。对于受管域，您可能需要修改受管域配置集；对于独立服务器，您可能需要删除 `/profile=default`。

```
/profile=default/subsystem=web/virtual-server=default-host:write-attribute(name=enable-welcome-root,value=false)
```
- 配置您的 Web 应用程序使用根上下文。**
要配置您的 Web 应用程序将根上下文 (/) 作为 URL 地址，请修改 `META-INF/` 或 `WEB-INF/` 目录下的 `jboss-web.xml`。用类似于下面的内容替换 `<context-root>` 指令。

```
<jboss-web>
  <context-root>/</context-root>
</jboss-web>
```
- 部署您的应用程序**
部署您的应用程序到服务器组或在第一个步骤修改的服务器里。这个应用程序应该可以通过这个地址访问：`http://SERVER_URL:PORT/`。

[提交 bug 报告](#)

第 18 章 WEB SERVICES 子系统

18.1. 配置 WEB SERVICES 选项

要配置 Web Services 选项，请点击 **Web Services** 菜单条目。下表解释了这些选项。

表 18.1. Web Services 的配置选项

选项	描述	CLI 命令
Modify WSDL Address	指定应用程序是否可以修改 WSDL 地址。默认值是 true 。	<pre>/profile=full- ha/subsystem=webserv ices/:write- attribute(name=modif y-wsdl- address,value=true)</pre>
WSDL Host	JAX-WS Web Service 的 WSDL 合约包括 <code><soap:address></code> 元素，它指向端点的位置。如果 <code><soap:address></code> 的值是一个有效的 URL，除非设置 modify-wsdl-address 为 true ，它不会被覆盖。如果 <code><soap:address></code> 的值不是一个有效的 URL，它会用 wsdl-host 和 wsdl-port 或 wsdl-secure-port 的值进行覆盖。如果 wsdl-host 被设置为 jbossws.undefined.host ，覆盖 <code><soap:address></code> 时请求者的主机地址将被使用。它的默认值是 \${jboss.bind.address:127.0.0.1} ，当 JBoss EAP 6 启动时如果没有指定绑定地址，将使用 127.0.0.1 。	<pre>/profile=full- ha/subsystem=webserv ices/:write- attribute(name=wsdl- host,value=127.0.0.1)</pre>
WSDL Port	用于重写 SOAP 端口的非安全端口。如果为 0 （默认值），端口将通过查询安装的连接器列表来确定。	<pre>/profile=full- ha/subsystem=webserv ices/:write- attribute(name=wsdl- port,value=80)</pre>
WSDL Secure Port	用于重写 SOAP 端口的安全端口。如果为 0 （默认值），端口将通过查询安装的连接器列表来确定。	<pre>/profile=full- ha/subsystem=webserv ices/:write- attribute(name=wsdl- secure- port,value=443)</pre>



注意

您可能需要修改配置集以使用不同的受管域配置集，或者从用于独立服务器的命令删除 `/profile=full-ha` 部分。

Web Services 子系统

要启用 Apache CXF 的日志，请在 `standalone/domain.xml` 文件里配置下列系统属性：

```
<system-properties>
<property name="org.apache.cxf.logging.enabled" value="true"/>
</system-properties>
```

[提交 bug 报告](#)

第 19 章 HTTP 群集和负载均衡

19.1. 介绍

19.1.1. 关于高可用性和负载均衡群集

群集 (Clustering) 指的是使用多个资源 (如服务器)，就好像单个实体一样。两个主要的群集类型是 **负载均衡 (Load balancing, LB)** 群集和 **高可用性 (High-availability, HA)** 群集。在 LB 群集里，所有的资源都同时运行，管理层对工作负荷进行分配。

在 HA 群集里，当运行时某个资源，如果第一个不可用，将选择另外一个可用的资源。HA 群集的目的是降低硬件、软件或网络故障的影响。

JBoss EAP 6 支持不同级别的群集。高可用性的运行时组件和应用程序是：

- 应用服务器实例
- 和内部 JBoss Web 服务器、Apache HTTP 服务器、Microsoft IIS 或 Oracle iPlanet Web 服务器一起使用的 Web 应用程序。
- Stateful、stateless 和 Entity Enterprise JavaBeans (EJB)
- 单点登录 (Single Sign On, SSO) 机制
- 分布式缓存
- HTTP 会话
- JMS 服务和消息驱动 Bean (Message-driven beans, MDB)

JBoss EAP 6 通过两个子系统使用群集：**jgroups** 和 **modcluster**。**ha** 和 **full-ha** 配置集都启用了这些子系统。在 JBoss EAP 6 里，这些服务按需要来启动和关闭，但只有配置为 **distributable** 的应用程序部署在服务器上才会启动这些服务。

在 JBoss EAP 6 里 Infinispan 是作为缓存供应商提供的。Infinispan 为 JBoss EAP 6 管理群集和复制缓存。

[提交 bug 报告](#)

19.1.2. 可从高可用性受益的组件

高可用性 (HA) 在 JBoss EAP 6 里分成更广的几个类别。

容器

JBoss EAP 6 (作为独立服务器运行) 的几个实例或服务器组的成员 (作为受管域的一部分运行) 可以配置为高可用的。这表示如果某个实例或成员停止或从群集消失，它的工作负荷将移至另外一个节点。工作负荷也可以通过这种方式提供负载均衡，所以具有更多、更好资源的服务器或服务器组可以承担更大部分的工作负荷，或者在高负荷时添加额外的能力。

Web 服务器

Web 服务器自身就可以使用某种兼容的负载均衡机制进行 HA 群集。最灵活的方式是 **mod_cluster** 连接器，它和 JBoss EAP 6 容器紧密地集成在一起。其他选择包含 Apache **mod_jk**、**mod_proxy** 连接器或 ISAPI 和 NSAPI 连接器。

应用程序

根据 Java EE 6 规格，部署的应用程序也可以是高可用性的。Stateless 或 stateful session EJB 可以进行群集，如果某个节点出现故障，另外一个节点可以接管。对于 stateful session bean，状态可以保持。

[提交 bug 报告](#)

19.1.3. HTTP 连接器概述

JBoss EAP 6 可以使用构建到外部 web 服务器的负载平衡和高可用性机制，如 Apache Web Server, Microsoft IIS 和 Oracle iPlanet。JBoss EAP 6 使用 HTTP 连接器和外部的 Web 服务器通讯。这些 HTTP 连接器是在 JBoss EAP 6 的 web 子系统里进行配置的。

Web 服务器包含了控制 HTTP 请求路由至 JBoss EAP 6 工作节点的途径的软件模块。每个模块的工作模式和配置都有所不同。您可以配置这些模块在多个 JBoss EAP 6 服务器节点间平衡负载，或在发生故障时将负荷移至其他服务器。这两种能力分别被称为**负载平衡 (Load Balancing)** 和**高可用性 (High Availability, HA)**。

JBoss EAP 6 支持几种不同的 HTTP 连接器。您要根据使用的 Web 服务器和需要的其他功能来进行选择。

下表列出了兼容 JBoss EAP 6 的 HTTP 连接器的不同之处。关于被支持的 HTTP 连接器的最新信息，请参考 <https://access.redhat.com/site/articles/111663>。

表 19.1. HTTP 连接器功能和约束

连接器	Web 服务器	支持的操作系统	支持的协议	适用于部署状态	支持会话关联
mod_cluster	JBoss Enterprise Web Server HTTPD、操作系统 (Red Hat Enterprise Linux, Hewlett-Packard HP-UX) 提供的 HTTPD	Red Hat Enterprise Linux, Microsoft Windows Server, Oracle Solaris, Hewlett-Packard HP-UX	HTTP, HTTPS, AJP	是。检测应用程序的部署和卸载并根据应用程序是否部署在某个服务器上动态决定是否将请求指引到该服务器上。	是
mod_jk	JBoss Enterprise Web Server HTTPD、操作系统 (Red Hat Enterprise Linux, Hewlett-Packard HP-UX) 提供的 HTTPD	Red Hat Enterprise Linux, Microsoft Windows Server, Oracle Solaris, Hewlett-Packard HP-UX	AJP	否。不管应用程序的状态，只要容器可用，就将请求指引到这个容器。	是

连接器	Web 服务器	支持的操作系统	支持的协议	适用于部署状态	支持会话关联
mod_proxy	JBoss Enterprise Web Server HTTPD	Red Hat Enterprise Linux\Microsoft Windows Server、Oracle Solaris	HTTP, HTTPS, AJP	否。不管应用程序的状态，只要容器可用，就将请求指引到这个容器。	是
ISAPI	Microsoft IIS	Microsoft Windows Server	AJP	否。不管应用程序的状态，只要容器可用，就将请求指引到这个容器。	是
NSAPI	Oracle iPlanet Web Server	Oracle Solaris	AJP	否。不管应用程序的状态，只要容器可用，就将请求指引到这个容器。	是

了解每个 HTTP 连接器的更多信息

- 第 19.5.1 节 “关于 mod_cluster HTTP 连接器”
- 第 19.6.1 节 “关于 Apache mod_jk HTTP 连接器”
- 第 19.7.1 节 “关于 Apache mod_proxy HTTP 连接器”
- 第 19.8.1 节 “关于 Internet Server API (ISAPI) HTTP 连接器”
- 第 19.9.1 节 “关于 Netscape Server API (NSAPI) HTTP 连接器”

JBoss EAP 6 支持的配置：<https://access.redhat.com/site/articles/111663>。

[提交 bug 报告](#)

19.1.4. 工作节点

HTTP 连接器节点

工作节点 (*worker node*)，有时候简称为节点，是从一个或多个面向客户的 HTTPD 服务器接受请求的 JBoss EAP 6 服务器。JBoss EAP 6 可以从自己的 HTTPD、JBoss EAP 6 附带的 HTTPD、Apache HTTPD、Microsoft IIS 或 Oracle iPlanet Web Server (以前是 Netscape Web Server) 接受请求。

关于 JBoss EAP 6 支持的 HTTP 连接器概述以及如何进行配置的信息，请参考 [第 19.1.3 节 “HTTP 连接器概述”](#)。

群集节点

群集节点 (*Cluster Node*) 是服务器群集里的成员。群集可以是负载均衡群集、高可用性群集，或者两者皆是。在负载均衡群集里，中央管理者根据不同的情况将负荷均匀地分摊到节点上。在高可用性群集里，某些节点积极运行，而其他节点则进行等待，当活动节点离开群集时开始运行。

[提交 bug 报告](#)

19.2. 连接器配置

19.2.1. 为 JBoss EAP 6 里的 HTTP 连接器定义线程池

概述

JBoss EAP 6 里的线程池可以通过 **Executor** 模型在不同组件间共享。这些池不仅可以由不同的 HTTP 连接器共享，也可以被 JBoss EAP 6 里支持 **Executor** 模型的其他组件共享。通过 HTTP 连接器线程池来满足当前的 Web 性能要求是件难办的事，它要求密切监控当前的线程池以及于其的负载需求。在本节里，您会学习如何通过 **Executor** 模型为 HTTP 连接器设立线程池。您也会学习通过命令行界面或编辑 XML 配置文件来进行设置。

过程 19.1. 为 HTTP 连接器设立线程池

1. 定义线程工厂

打开配置文件 **standalone.xml** (独立服务器) 或 **domain.xml** (受管域)。这个文件位于 **EAP_HOME/standalone/configuration** 或 **EAP_HOME/domain/configuration** 目录。

添加下列子系统条目，按照您的服务器的需要修改相关的值。

```
<subsystem xmlns="urn:jboss:domain:threads:1.0">
  <thread-factory name="http-connector-factory" thread-name-
    pattern="HTTP-%t" priority="9" group-name="uq-thread-pool"/>
</subsystem>
```

如果您想用 CLI 来完成这个任务，请在 CLI 命令行提示下执行下列命令：

```
[standalone@localhost:9999 /] ./subsystem=threads/thread-
factory=http-connector-factory:add(thread-name-pattern="HTTP-%t",
priority="9", group-name="uq-thread-pool")
```

2. 创建执行器 (Executor)

您可以使用六种内置的 **Executor** 类来充当这个工厂的执行器。这六种执行器是：

- **unbounded-queue-thread-pool**：这种类型的线程池总是接受任务。如果少于线程最大数目的线程在运行，新的线程将启动以运行提交的任务；否则，任务将放入非绑定的 FIFO 队列，等待有可用的线程再被执行。



注意

Executors.singleThreadExecutor() 提供的 **single-thread executor** 类型基本上是一个具有单个线程限制 **unbounded-queue** 执行器。这个类型的执行器是用 **unbounded-queue-thread-pool-executor** 元素进行部署的。

- **bounded-queue-thread-pool**：这个类型的执行器维护了一个固定长度的队列及两个池大小参数：**core** 和 **maximum**。当接受任务时，如果运行的池线程小于 **core**，新的线程将启动以执行这个任务。如果大于 **Core** 且队列里有空位，任务将放入队列。如果队列没有空位且运行的池线程小于 **maximum**，新的线程将启动以执行这个任务。如果执行器启用了阻塞模式，调用线程将阻塞至队列里出现空位。如果配置了 **handoff** 执行器，任务将委托给 **handoff** 执行器。否则，任务将被拒绝。
- **blocking-bounded-queue-thread-pool**：带有绑定队列的线程池执行器，队列里提交任务的线程可能会阻塞。这样的线程池也有 **Core** 和 **Maximum** 参数及指定的队列长度。当任务被提交时，如果运行线程的数目小于 **Core** 参数，那么新的线程将被创建。如果大于 **Core**

且队列里还有空位，任务将进行排队。而如果队列里没有空位且运行线程的数目小于 **Maximum** 参数，新的线程将被创建。

- **queueless-thread-pool**：有时我们需要简单的线程池在单独的线程里执行任务，当它们完成任务后还可以重用而无需中介队列。既然任务总是在接受后立即执行而不是先接受任务然后延迟执行至其他运行的任务已完成，处理长期运行的任务时这种类型的池比较合适，如利用阻塞的 I/O。这种类型的执行器是通过 **queueless-thread-pool-executor** 元素声明的。
- **blocking-queueless-thread-pool**：没有队列的线程池执行器，队列里的提交任务的线程可能阻塞。当任务被提交时，如果运行的线程数小于 **Maximum** 参数，新的线程将被创建。否则，调用将阻塞直至另外一个线程完成它的任务并接受新任务。
- **scheduled-thread-pool**：这是一个特殊类型的执行器，它的目的是基于 **java.util.concurrent.ScheduledThreadPoolExecutor** 类在专门的时间和时间间隔执行任务。这种类型的执行器是用 **scheduled-thread-pool-executor** 元素来配置的：

在这个例子里，我们将使用 **unbounded-queue-thread-pool** 来充当执行器。修改 **max-threads** 和 **keepalive-time** 参数的值来满足您的服务器的需要。

```
<unbounded-queue-thread-pool name="uq-thread-pool">
  <thread-factory name="http-connector-factory" />
  <max-threads count="10" />
  <keepalive-time time="30" unit="seconds" />
</unbounded-queue-thread-pool>
```

或者您更愿意使用 CLI：

```
[standalone@localhost:9999 /] ./subsystem=threads/unbounded-queue-
thread-pool=uq-thread-pool:add(thread-factory="http-connector-
factory", keepalive-time={time=30, unit="seconds"}, max-threads=30)
```

3. 让 HTTP 连接器使用这个线程池

在相同的配置文件里，找到 **web** 子系统下的 HTTP 连接器元素并进行修改，让它使用之前步骤里定义的线程池。

```
<connector name="http" protocol="HTTP/1.1" scheme="http" socket-
binding="http" executor="uq-thread-pool" />
```

或者您更愿意使用 CLI：

```
[standalone@localhost:9999 /] ./subsystem=web/connector=http:write-
attribute(name=executor, value="uq-thread-pool")
```

4. 重启服务器

重启服务器（独立服务器或受管域）让修改可以生效。请用下列 CLI 命令来确认上面步骤里的修改已经生效：

```
[standalone@localhost:9999 /] ./subsystem=threads:read-
resource(recursive=true)
{
  "outcome" => "success",
```

```

"result" => {
  "blocking-bounded-queue-thread-pool" => undefined,
  "blocking-queueless-thread-pool" => undefined,
  "bounded-queue-thread-pool" => undefined,
  "queueless-thread-pool" => undefined,
  "scheduled-thread-pool" => undefined,
  "thread-factory" => {"http-connector-factory" => {
    "group-name" => "uq-thread-pool",
    "name" => "http-connector-factory",
    "priority" => 9,
    "thread-name-pattern" => "HTTP-%t"
  }},
  "unbounded-queue-thread-pool" => {"uq-thread-pool" => {
    "keepalive-time" => {
      "time" => 30L,
      "unit" => "SECONDS"
    },
    "max-threads" => 30,
    "name" => "uq-thread-pool",
    "thread-factory" => "http-connector-factory"
  }}
}
}
[standalone@localhost:9999 /] ./subsystem=web/connector=http:read-
resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "configuration" => undefined,
    "enable-lookups" => false,
    "enabled" => true,
    "executor" => "uq-thread-pool",
    "max-connections" => undefined,
    "max-post-size" => 2097152,
    "max-save-post-size" => 4096,
    "name" => "http",
    "protocol" => "HTTP/1.1",
    "proxy-name" => undefined,
    "proxy-port" => undefined,
    "redirect-port" => 443,
    "scheme" => "http",
    "secure" => false,
    "socket-binding" => "http",
    "ssl" => undefined,
    "virtual-server" => undefined
  }
}

```

结果

您已经创建一个线程工厂和执行器，而且修改了 HTTP 连接器来使用这个线程池。

[提交 bug 报告](#)

19.3. WEB SERVER 配置

19.3.1. 关于独立 Apache HTTP 服务器

JBoss EAP 6 使用了 Red Hat 企业版 6 认证版本包含的 Apache HTTP 服务器进行测试和支持。Apache HTTP 服务器也可用于其他配置，如 Microsoft Windows Server。然而，既然 Apache HTTP 服务器是 Apache Foundation 创建的一个独立项目，确认用户使用的 Apache HTTP 服务器版本是否和 JBoss EAP 兼容通常是很困难的。

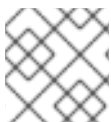
独立的 Apache HTTP 服务器现在是 JBoss EAP 6 附带的可单独下载的软件包。这简化了非 Red Hat 企业版 Linux 环境、或者已经配置了 HTTP 服务器但 Web 应用程序想使用单独实例的情况下的安装和配置。您可以从客户服务门户单独下载这个 HTTP 服务器，它位于安装平台对应的 JBoss 的 EAP 6 版本下。

[提交 bug 报告](#)

19.3.2. 安装 JBoss EAP 6 附带的 Apache HTTP 服务器（ZIP 方式）

预备条件

- 根用户或管理员权限。
- 被支持的 Java 版本。
- 安装下列软件包：
 - krb5-workstation
 - mod_auth_kerb
 - elinks（apachectl 功能所必需的）
- 您必须安装 Apache Portability Runtime（APR）。在 Red Hat 企业版 Linux 里，请安装 **apr-util-devel**。



注意

在 Red Hat 企业版 Linux 7 上，LDAP 验证要求安装 **apr-util-ldap**。



注意

关于 Microsoft Windows 服务器环境下安装 Apache HTTP 服务器的信息，请参考《JBoss Enterprise Web Server 2 安装指南》里『Installing Enterprise Web Server on Windows』章节的『Configuring the Environment』。

过程 19.2. 安装 Apache HTTP 服务器

1. 在 Red Hat 客户门户找到对应您的平台的 JBoss EAP 下载列表。
登录到 Red Hat 客户服务门户 <https://access.redhat.com>。点击 **Downloads**，然后选择 **Product Downloads** 列表里的 **Red Hat Enterprise Application Server**。从 **Version** 下拉菜单里选择正确的 JBoss EAP 版本。
2. 从列表里选择 HTTP Binary。
找到对应您的操作系统和架构的 **Apache HTTP Server** 选项。点击 **Download** 链接，下载包含 Apache HTTP 服务器的 ZIP 文件到本地主机。
3. 解压 ZIP 到要运行 Apache HTTP 服务器的位置。

在您首选的服务器上解压 ZIP 文件到一个临时位置。ZIP 文件将创建 *jboss-ews-version-number* 下的 **httpd** 目录。复制 **httpd** 文件夹并将其放在您要安装 JBoss EAP 6 的目录里，这个目录通常被称为 **EAP_HOME**。

您的 Apache HTTP 服务器现在位于 **EAP_HOME/httpd/** 目录。正如其他 JBoss EAP 6 文档里找到的那样，您现在可以用 **HTTPD_HOME** 表示这个位置。

4. 运行 Post-Installation 脚本并创建 apache 用户和组帐号

在终端窗口里，切换到根用户帐号，进入 **EAP_HOME/httpd** 目录并执行下列命令。

```
./postinstall
```

然后，通过下列命令检查是否存在名为 **apache** 的用户：

```
id apache
```

如果这个用户不存在，那您需要添加它及合适的用户组。为此，请执行下列命令：

```
/usr/sbin/groupadd -g 91 -r apache 2> /dev/null || :  
/usr/sbin/useradd -c "Apache" -u 48 -g 91 -s /sbin/nologin -r apache  
2>  
/dev/null || :
```

执行这个命令后，如果 **apache** 用户将运行 HTTPD 服务器，你需要修改 HTTP 目录的所有权：

```
chown -R apache:apache httpd
```

要测试上述命令是否执行成功，请检查 **apache** 用户是否具有对 Apache HTTP 服务器安装路的执行权限。

```
ls -l
```

输出结果应该类似于：

```
drwxrwxr-- 11 apache apache 4096 Feb 14 06:52 httpd
```

5. 配置 Apache HTTP 服务器。

用下列命令切换到新的用户帐号

```
sudo su apache
```

并配置 Apache HTTP 服务器为 **apache** 用户来满足机构的需要。您可以参考 Apache Foundation 的文档：<http://httpd.apache.org/>。

6. 启动 Apache HTTP 服务器。

用下列命令启动 Apache HTTP 服务器：

```
EAP_HOME/httpd/sbin/apachectl start
```

7. 停止 Apache HTTP 服务器。

用下列命令停止 Apache HTTP 服务器：

```
EAP_HOME/httpd/sbin/apachectl stop
```

[提交 bug 报告](#)

19.3.3. 在 Red Hat 企业版 Linux 5/6/7 (RHEL) 里用 RPM 方式安装 Apache HTTP 服务器

预备条件

- 根用户或管理员权限。
- 被支持的 Java 版本。
- 最新版本的 `elinks` 软件包 (`apachectl` 功能所必需的)。
- 订阅 Red Hat Enterprise Linux (RHEL) 频道 (从 RHEL 频道安装 Apache HTTP 服务器)。
- 订阅 `jbossplatform-6-ARCH-server-VERS-rpm` Red Hat Network (RHN) 频道 (安装 EAP 专有的 Apache HTTP 服务器)。

你可以用下列方法安装 Apache HTTP 服务器：

- 通过 RHEL 频道：要安装 Apache HTTP 服务器，活动的 Red Hat Enterprise Linux (RHEL) 频道的订阅是必需的。
- 通过 `jbossplatform-6-ARCH-server-VERS-rpm` 频道 (JBoss EAP 专有的版本)：JBoss EAP 发布了自己的 Apache HTTP 服务器版本。要安装专有的 Apache HTTP 服务器，活动的 `jbossplatform-6-ARCH-server-VERS-rpm` 频道的订阅是必需的。

过程 19.3. 在 Red Hat 企业版 Linux 5/6 (RHEL) 里用 RPM 方式安装和配置 Apache HTTP 服务器

1. Install httpd

要安装 JBoss EAP 专有的 `httpd` 软件包版本，请运行下列命令：

```
yum install httpd
```

要从 Red Hat Enterprise Linux (RHEL) 频道显性地安装 `httpd` 软件包，请运行下列命令：

```
yum install httpd --disablerepo=jbossplatform-6-*
```



注意

您必须运行上述命令之一来安装 `httpd` 软件包。

2. 设置服务引导行为

您可以通过命令行或服务配置图形化工具定义 `httpd` 服务在引导时的行为。请运行下列命令来定义行为：

```
chkconfig httpd on
```

要使用服务配置工具，请运行下列命令并在显示的窗口里修改服务设置：

```
system-config-services
```

3. 启动 httpd

用下列命令启动 **httpd**：

```
service httpd start
```

4. 停止 httpd

用下列命令停止 **httpd**：

```
service httpd stop
```

过程 19.4. 在 Red Hat 企业版 Linux 7 里用 RPM 方式安装和配置 Apache HTTP 服务器

1. 安装 httpd22

要安装 JBoss EAP 专有版本的 **httpd22** 软件包，请运行下列命令：

```
yum install httpd22
```

2. 设置服务引导行为

运行下列命令在引导时启动 **httpd22** 服务：

```
systemctl enable httpd22.service
```

3. 启动 httpd22

用下列命令启动 **httpd22**：

```
systemctl start httpd22.service
```

4. 停止 httpd22

用下列命令停止 **httpd22**：

```
systemctl stop httpd22.service
```

[提交 bug 报告](#)

19.3.4. httpd 上的 mod_cluster 配置

介绍

mod_cluster 是一个基于 HTTPD 的负载均衡器。它使用一个通讯频道从 HTTPD 转发请求到应用服务器节点。您可以通过下面的 Derivative 配置 HTTPD 上的 **mod_cluster**。



注意

您不需要使用 **ProxyPass** directive，因为 **mod_cluster** 会自动配置必须转发给 JBossWEB 的 URL。

表 19.2. **mod_cluster** Derivatives

Derivative	描述	值
CreateBalancers	定义如何在 HTTPD 虚拟主机里创建负载均衡器。它允许这样的 directive : ProxyPass /balancer://mycluster1/ 。	<p>0: 创建 httpd 里定义的所有虚拟主机</p> <p>1: 不创建负载均衡器 (定义平衡器名称要求至少一个 ProxyPass 或 ProxyMatch)</p> <p>2: 只创建主服务器</p> <p>默认值 : 2</p> <p>使用这个值时, 请不要忘记配置 ProxyPass directive 里的平衡器, 因为默认值是一个空的 Sticky Session, nofailover=Off 和通过 MCMP CONFIG 消息接收的值都会被忽略。</p>
UseAlias	检查别名是否对应服务器名。	<p>0: 忽略别名</p> <p>1: 检查别名</p> <p>默认值 : 0</p>
LBstatusRecalTime	重新计算节点状态的负载均衡逻辑的时间间隔 (秒)。	默认值 : 5 秒钟
WaitForRemove	在已删除的节点被 HTTPD 忘记前的时间 (秒)。	默认值 : 10 秒
ProxyPassMatch/ProxyPass	<p>ProxyPassMatch 和 ProxyPass 是 mod_proxy directive, 当使用 ! (而非 back-end url) 时阻止路径里的 reverse-proxy。它用于允许 HTTPD 服务于静态信息, 如图像。例如 :</p> <p>ProxyPassMatch ^(/.*\..gif)\$!</p> <p>上面的例子允许 HTTPD 直接保存 .gif 文件。</p>	

mod_cluster 逻辑里的 hot-standby 节点是当所有其他节点都下线时, 所有请求去往的最后的资源节点。这和 mod_proxy 里的 hot-standby 逻辑类似。

要配置 hot-standby 节点, 用 factor 为 0 的 simple-load-provider 替换 mod_cluster 子系统里的 dynamic-load-provider, 例如 :

```
<subsystem xmlns="urn:jboss:domain:modcluster:1.2">
  <mod-cluster-config advertise-socket="modcluster" connector="ajp">
    -   <dynamic-load-provider>
    -     <load-metric type="busyness"/>
    -   </dynamic-load-provider>
    +   <simple-load-provider factor="0"/>
  </mod-cluster-config>
</subsystem>
```

```
</mod-cluster-config>
</subsystem>
```

在 `mod_cluster-manager` 控制台里，节点状态显示为 OK 且负载为 0。更多信息请参考《JBoss Enterprise Application Platform 开发指南》里的『*Apache mod_cluster-manager Application*』章节。

例如，如果有 3 个节点：

- 节点 A，负载：10
- 节点 B，负载：10
- 节点 C，负载：0

负载将在节点 A 和 B 之间进行平衡。如果两者都不可用，节点 C 将承担负载。

mod_manager

除非特别说明，在所有情况下，`mod_manger directive` 的上下文都是 `VirtualHost`。`server config` 上下文暗示 `directive` 必须位于 `VirtualHost` 配置的外面。如果没有，则会显示错误信息且不会启动 HTTPD。

表 19.3. `mod_manager` Derivatives

Derivative	描述	值
EnableMCPMReceive	允许 VirtualHost 从节点接收 MCPM。HTTPD 配置文件里包含的 EnableMCPMReceive 可以允许 mod_cluster 运行。配置 Advertise 时请将它保存在 VirtualHost 里。	
MemManagerFile	mod_manager 用来保存配置、为共享内存生成密钥或锁定文件的基础名称。它必须是一个绝对路径名；如果需要可以创建 directive。我们推荐将这些文件置于本地驱动而非 NFS 共享目录里。 Context: server config	\$server_root/logs/
Maxcontext	mod_cluster 支持的上下文的最大数量 Context: server config	默认值：100
Maxnode	mod_cluster 支持的节点的最大数量 Context: server config	默认值：20
Maxhost	mod_cluster 支持的主机（别名）的最大数量。它也包含平衡器的最大数目。 Context: server config	10

Derivative	描述	值
Maxsessionid	<p>保存的活动 sessionid 的数量，它提供 mod_cluster-manager 处理程序里活动会话的数量。当 mod_cluster 在 5 分钟内没有从会话里接收信息时，会话将变成不活动的。</p> <p>Context: server config</p> <p>这个字段只适用于演示和调试目的。</p>	0：逻辑没被激活。
MaxMCMPMaxMessSize	来自其他 Max Directive 的 MCMP 消息的最大尺寸	从其他 Max Directive 进行计算。 最小值：1024
ManagerBalancerName	当 JBoss AS/JBossWeb/Tomcat 没有提供平衡器名称时使用的平衡器名称。	mycluster
PersistSlots	<p>告诉 mod_slotmem 将节点、别名和上下文保存到文件里。</p> <p>Context: server config</p>	Off
CheckNonce	当使用 mod_cluster-manager 处理程序时切换 Nonce 检查	on/off 默认：on - Nonce 检查
AllowDisplay	在 mod_cluster-manager 主页面上切换其他显示信息。	on/off 默认：off - 只显示版本号
AllowCmd	允许使用 mod_cluster-manager URL 的命令。	on/off 默认：on - 允许的命令
ReduceDisplay	减少显示在主 mod_cluster-manager 页面上的信息，所以在这个页面上可以显示更多的节点。	on/off 默认：off - 显示全部信息

Derivative	描述	值
SetHandler mod_cluster-manager	<div><div>显示 mod_cluster 从群集里看到的节点的信息。这些信息包括普通信息活动会话的数量。</div><div><pre><Location /mod_cluster- manager> SetHandler mod_cluster-manager Order deny,allow Allow from 127.0.0.1 </Location></pre></div></div>	<div>on/off</div> <div>默认值：off</div>



注意

当访问 httpd.conf 里定义的位置时：

Transferred：对应发送到后台服务器的 POST 数据。

Connected：对应请求 mod_cluster 状态页面时已被处理的请求的数量。

Num_sessions：对应 mod_cluster 报告为活动的会话的数量（过去 5 分钟有请求）。如果 Maxsessionid 为 0，这个字段不会出现。这个字段只适用于演示和调试目的。

[提交 bug 报告](#)

19.3.5. 将外部 Web 服务器用作 JBoss EAP 6 应用程序的 Web 前端

概述

关于将外部 Web 服务器用作 Web 前端的原因，以及 JBoss EAP 6 支持的不同 HTTP 连接器的优缺点，请参考 [第 19.1.3 节“HTTP 连接器概述”](#)。在某些情况下，您可以使用操作系统附带的 Apache HTTP 服务器。否则，您可以使用作为 JBoss Enterprise Web Server 一部分发行的 Apache HTTP 服务器。

在您已决定使用哪个 Web 服务器和 HTTP 连接器后，请参考下列过程之一：

- [第 19.3.2 节“安装 JBoss EAP 6 附带的 Apache HTTP 服务器（ZIP 方式）”](#)
- [第 19.5.3 节“安装 mod_cluster 模块至 Apache HTTP 服务器或 JBoss Enterprise Web Server（ZIP 方式）”](#)
- [第 19.6.3 节“安装 mod_jk 模块到 Apache HTTP 服务器（ZIP 方式）”](#)
- [第 19.8.3 节“配置 Microsoft IIS 使用 ISAPI Redirector”](#)
- [第 19.9.2 节“在 Oracle Solaris 上配置 NSAPI Connector”](#)

- 第 19.3.6 节 “配置 JBoss EAP 6 接受外部 Web 服务器的请求”

提交 bug 报告

19.3.6. 配置 JBoss EAP 6 接受外部 Web 服务器的请求

概述

JBoss EAP 6 不需要直到它从中接受请求的代理，只需要直到端口和协议。但 `mod_cluster` 不是这样的，它和 JBoss EAP 6 的配置耦合得更紧密。下列步骤适用于 `mod_jk`、`mod_proxy`、`ISAPI` 和 `NSAPI`。请用自己的设置替换这个例子里的协议和端口。

要配置 JBoss EAP 6 的 `mod_cluster`，请参考 第 19.5.6 节 “配置 `mod_cluster` 工作节点”。

必须具备的条件

- 您需要登录管理 CLI 或管理控制台来执行这个任务。本节使用管理 CLI，但在管理控制台里基本步骤是一样的。
- 您需要一个所使用协议的列表，如 HTTP、HTTPS 或 AJP。

过程 19.5. 编辑配置并添加套接字绑定

1. 配置 `jvmRoute` 系统属性。

在默认情况下，`jvmRoute` 会被设置为服务器名称。如果您需要定制，您可以使用下列命令。根据您使用的配置集或独立服务器，替换或删除命令的 `/profile=ha` 部分。用你自己的 `jvmRoute` 替换 `CUSTOM_ROUTE_NAME`。

```
[user@localhost:9999 /] /profile=ha/subsystem=web:write-attribute(name="instance-id",value="CUSTOM_ROUTE_NAME")
```

2. 列出 Web 子系统可用的连接器。



注意

这个步骤只有在使用独立服务器的 `standalone-ha.xml` 配置或受管域的 `ha` 或 `full-ha` 配置集时才需要。这些配置已经包含了必需的连接器。

为了使外部的 Web 服务器可以连接 JBoss EAP 6 的 Web 服务器，Web 子系统需要一个连接器。每个协议都需要自己的连接器，它绑定到套接字组上。

要列出当前可用的连接器，请使用下列命令：

```
[standalone@localhost:9999 /] /subsystem=web:read-children-names(child-type=connector)
```

如果没有指定您需要的连接器（如 HTTP、HTTPS、AJP），您需要添加连接器。

3. 读取连接器的配置。

要查看连接器配置的细节，您可以读取它的配置。下列命令读取 AJP 连接器的配置。其他连接器具有类似的配置输出。

```
[standalone@localhost:9999 /] /subsystem=web/connector=ajp:read-
```

```
resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "enable-lookups" => false,
    "enabled" => true,
    "max-post-size" => 2097152,
    "max-save-post-size" => 4096,
    "protocol" => "AJP/1.3",
    "redirect-port" => 8443,
    "scheme" => "http",
    "secure" => false,
    "socket-binding" => "ajp",
    "ssl" => undefined,
    "virtual-server" => undefined
  }
}
```

4. 将必需的连接器添加到 Web 子系统里。

要添加连接器到 Web 子系统里，您需要一个套接字绑定。这个套接字绑定应添加到您的服务器或服务器组使用的套接字绑定组里。下面的步骤假设您的服务器组是 **server-group-one**，套接字绑定组是 **standard-sockets**。

a. 添加套接字到套接字绑定组里。

要添加套接字到套接字绑定组，请执行下列命令（替换协议和端口）。

```
[standalone@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=ajp:add(port=8009)
```

b. 添加套接字绑定到 Web 子系统里。

执行下列命令将连接器添加到 Web 子系统里（替换套接字绑定名和协议）。

```
[standalone@localhost:9999 /]
/subsystem=web/connector=ajp:add(socket-binding=ajp,
protocol="AJP/1.3", enabled=true, scheme="http")
```

[提交 bug 报告](#)

19.4. 群集

19.4.1. Clustering 子系统使用 TCP 通讯

在默认情况下，群集节点用 UDP 协议监控其他节点的状态。某些网络只允许使用 TCP。在这种情况下，您可以在配置里添加 **TCPPING** 协议栈并将作为默认机制。基于命令行的管理 CLI 里也有这些配置选项。

mod_cluster 子系统默认也使用 UDP 通讯，但您可以选择使用 TCP。

参考下列两个过程配置 JGroups 和 **mod_cluster** 子系统以使用 TCP 网络通讯：

- [第 19.4.2 节“配置 JGroups 子系统使用 TCP”](#)
- [第 19.4.3 节“禁用 mod_cluster 子系统的广告”](#)

[提交 bug 报告](#)

19.4.2. 配置 JGroups 子系统使用 TCP

在默认情况下，JGroups 系统使用多点传送 UDP 通讯。使用下列过程来配置 JGroups 系统使用多点传送 TCP。

要配置 `mod_cluster` 子系统使用 TCP，请参考 [第 19.4.3 节“禁用 `mod_cluster` 子系统的广告”](#)。

1. 根据您的环境修改下列脚本。

复制下列脚本到文本编辑器里。如果您使用受管域里不同的配置集，请相应地修改配置集名称。如果您使用独立服务器，请删除命令行里的 `/profile=full-ha` 部分。像下面这样修改命令底部列出的属性。这些属性都是可选的。

`initial_hosts`

用逗号隔开的主机列表，里面的主机被认为是众所周知的，可用来查找初始成员资格。

`port_range`

如果需要，您可以分配一个端口范围。如果您分配的端口范围为 2，而初始端口为 7600，那么 TCPING 将试图连接每台主机的 7600-7601 端口。这个属性是可选的。

`timeout`

群集成员的可选超时时间（毫秒）。

`num_initial_members`

在群集被认为完成之前的节点数。这个属性是可选的。

```
batch
## If tcp is already added then you can remove it ##
/profile=full-ha/subsystem=jgroups/stack=tcp:remove
/profile=full-ha/subsystem=jgroups/stack=tcp:add(transport={"type"
=>"TCP", "socket-binding" => "jgroups-tcp"})
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-
protocol(type=TCPPING)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-
protocol(type=MERGE2)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-
protocol(type=FD_SOCKET, socket-binding=jgroups-tcp-fd)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=FD)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-
protocol(type=VERIFY_SUSPECT)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-
protocol(type=BARRIER)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-
protocol(type=pbcaster.NAKACK)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-
protocol(type=UNICAST2)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-
protocol(type=pbcaster.STABLE)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-
protocol(type=pbcaster.GMS)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=UFC)
```

```

/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=MFC)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-
protocol(type=FRAG2)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-
protocol(type=RSVP)
/profile=full-ha/subsystem=jgroups:write-attribute(name=default-
stack,value=tcp)
run-batch
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=initial_hos
ts/:add(value="HostA[7600],HostB[7600]")
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=port_range/
:add(value=0)
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=timeout/:ad
d(value=3000)
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=num_initial
_members/:add(value=3)

```

2. 以批处理方式运行脚本。



警告

运行这个配置集的服务器必须在执行批处理文件前先关闭。

在终端窗口里，进入包含 **jboss-cli.sh** 脚本的目录并输入下列命令。

```
./jboss-cli.sh -c --file=SCRIPT_NAME
```

这里的 **SCRIPT_NAME** 是名称及包含脚本的路径。

结果

TCPPING 栈对于 **JGroups** 子系统不可用。如果使用它，**JGroups** 子系统将对所有网络通讯使用 **TCP**。要配置 **mod_cluster** 子系统使用 **TCP**，请参考 [第 19.4.3 节“禁用 mod_cluster 子系统的广告”](#)。

[提交 bug 报告](#)

19.4.3. 禁用 mod_cluster 子系统的广告

在默认情况下，**mod_cluster** 子系统的平衡器使用多点传送 **UDP** 来广播其功能到后台 **Worker**。如果您愿意，您可以禁用广告。请用下列命令来配置这种行为。

过程 19.6.

1. 修改 HTTPD 配置。

修改 **HTTPD** 配置来禁用服务器广告并使用代理列表。代理列表在工作节点上配置，包含它可以对话的所有启用了 **mod_cluster** 的 **Web** 服务器。

Web 服务器的 `mod_cluster` 配置通常位于 `/etc/httpd/` 或 HTTPD 安装的 `etc/httpd/` 目录里（对于非标准安装）。关于这个文件的更多信息，请参考第 19.5.3 节“安装 `mod_cluster` 模块至 Apache HTTP 服务器或 JBoss Enterprise Web Server（ZIP 方式）”和第 19.5.5 节“为启用 `mod_cluster` 的 Web 服务器配置服务器的 `Advertisement` 属性”。打开包含侦听 MCPM 请求（使用 `EnableMCPMReceive` 指令）的虚拟主机的文件，并通过修改 `ServerAdvertise` 指令禁用服务器广告。

ServerAdvertise Off

2. 禁用 JBoss EAP 6 里的 `mod_cluster` 子系统的广告并提供一个代理列表。

通过基于 Web 的管理控制台或命令行管理 CLI，您可以禁用 `mod_cluster` 子系统的广告提供一个代理列表。因为如果禁用广告，`mod_cluster` 子系统无法自动发现代理，所以这个代理列表是必需的。

o 管理控制台

如果您使用受管域，您只可以在启用 `mod_cluster` 的配置集里配置它，如 `ha` 和 `full-ha` 配置集。

1. 登录到管理控制台并选择屏幕顶部的 **Configuration**。如果您使用的是受管域，请从左上角的 **Profiles** 下拉菜单里选择 `ha` 或 `full-ha` 配置集。
2. 展开 **Subsystems** 菜单，然后展开 **Web** 子菜单并选择 `mod_cluster`。
3. 点击 `mod_cluster` 下 **Advertising** 标签页里的 **Edit**。要禁用广告，清除 **Advertise** 旁边的复选框，然后点击 **Save**。

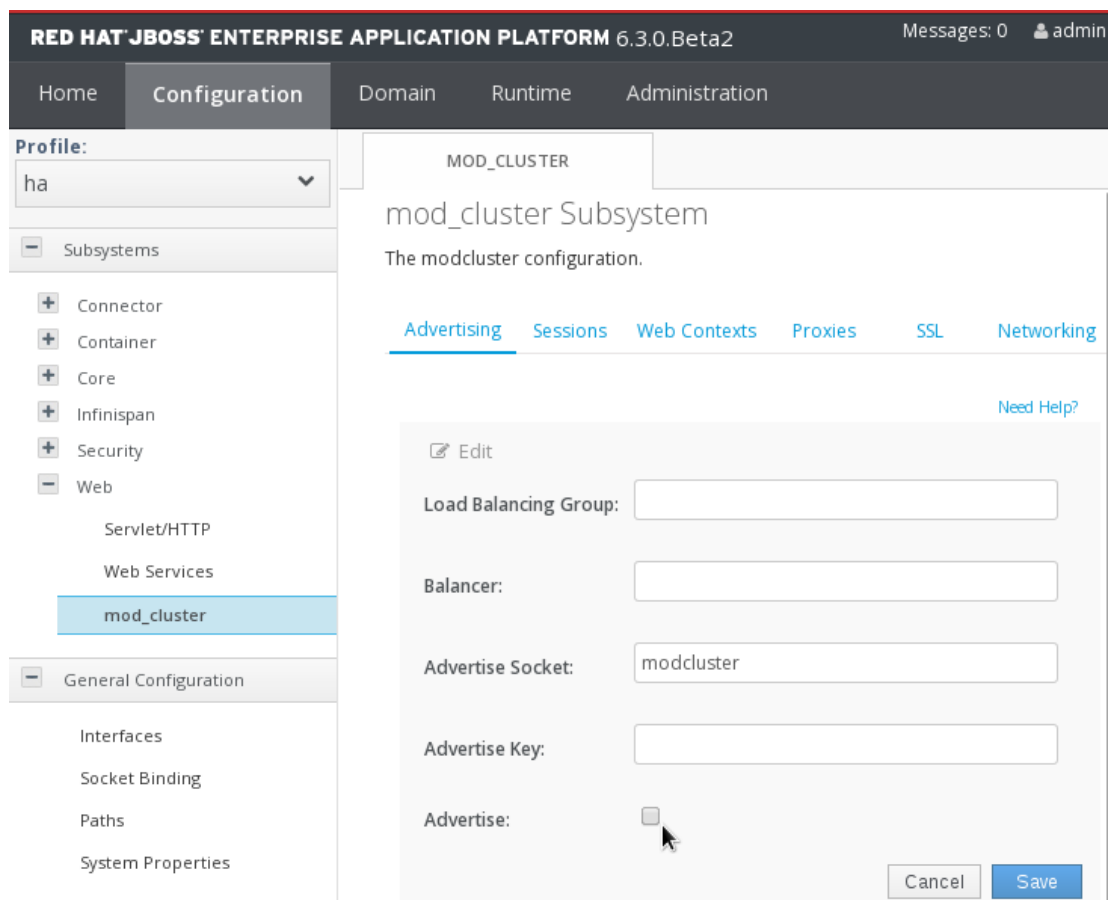


图 19.1. `mod_cluster` 广告配置屏幕

4. 点击的 **Proxies** 标签页。点击 **Edit** 并在 **Proxy List** 字段里输入代理服务器列表。正确的语法是逗号隔开的 **HOSTNAME:PORT** 字符串列表，如：

```
10.33.144.3:6666,10.33.144.1:6666
```

点击 **Save** 按钮。

管理 CLI

下面的两个管理 CLI 命令创建了和上面的管理控制台相同的配置。它们假设您运行的是受管域且您的服务器组使用了 **full-ha** 配置集。如果您使用了不同的配置，请在命令里修改名称。如果你使用了运行 **standalone-ha** 配置集的独立服务器，请从命令里删除 **/profile=full-ha**。

```
/profile=full-ha/subsystem=modcluster/mod-cluster-
config=configuration/:write-attribute(name=advertise,value=false)

/profile=full-ha/subsystem=modcluster/mod-cluster-
config=configuration/:write-attribute(name=proxy-
list,value="10.33.144.3:6666,10.33.144.1:6666")
```

结果

HTTPD 平衡器不再向工作节点广播且不再使用 UDP 多点传送。

[提交 bug 报告](#)

19.4.4. 为 HornetQ 群集切换 UDP 至 TCP

下面的例子使用了 EAP 6 附带的 **standalone-full-ha.xml** 默认配置文件。



注意

如果启用了安全性，您必须设置 **cluster-password** 属性：

```
<cluster-
password>${jboss.messaging.cluster.password:ChangeMe}</cluster-
password>
```

1. 删除 **broadcast-groups** 和 **discovery-groups**：

```
<broadcast-groups>
  <broadcast-group name="bg-group1">
    <socket-binding>messaging-group</socket-binding>
    <broadcast-period>5000</broadcast-period>
    <connector-ref>netty</connector-ref>
  </broadcast-group>
</broadcast-groups>
<discovery-groups>
  <discovery-group name="dg-group1">
    <socket-binding>messaging-group</socket-binding>
    <refresh-timeout>10000</refresh-timeout>
```

```
</discovery-group>
</discovery-groups>
```

2. 也可以选择删除 "messaging-group" 套接字绑定

```
<socket-binding name="messaging-group" port="0" multicast-
address="${jboss.messaging.group.address:231.7.7.7}" multicast-
port="${jboss.messaging.group.port:9876}"/>
```

3. 配置合适的 Netty 连接器 - 群集里的其他每个节点都对应一个。

例如，如果群集是三个节点则配置两个 Netty 连接器；如果群集是两个节点则配置一个 Netty 连接器。下面是关于三个节点群集的配置示例：

```
<netty-connector name="other-cluster-node1" socket-binding="other-
cluster-node1"/>
<netty-connector name="other-cluster-node2" socket-binding="other-
cluster-node2"/>
```

4. 配置相关的套接字绑定。



注意

如果需要，系统属性替换可以用于 "host" 或 "port"。

```
<outbound-socket-binding name="other-cluster-node1">
  <remote-destination host="otherNodeHostName1" port="5445"/>
</outbound-socket-binding>
<outbound-socket-binding name="other-cluster-node2">
  <remote-destination host="otherNodeHostName2" port="5445"/>
</outbound-socket-binding>
```

5. 配置 cluster-connection 使用这些连接而不是默认的 discovery-group：

```
<cluster-connection name="my-cluster">
  <address>jms</address>
  <connector-ref>netty</connector-ref>
  <static-connectors>
    <connector-ref>other-cluster-node1</connector-ref>
    <connector-ref>other-cluster-node2</connector-ref>
  </static-connectors>
</cluster-connection>
```

您必须为每个群集节点重复这个过程，这样每个节点都有对于群集里其他每个节点的连接器。



注意

请不要配置连接自身的节点。这会被当作配置错误。

[提交 bug 报告](#)

19.5. WEB、HTTP 连接器和 HTTP 群集

19.5.1. 关于 mod_cluster HTTP 连接器

mod_cluster 是在 JBoss Web 容器里启用负载平衡的模块。它被称为 *connector*。使用哪个连接器取决于您用于 JBoss EAP 6 的 Web 容器。要了解其他连接器的详情，请参考下面的内容：

- [第 19.6.1 节 “关于 Apache mod_jk HTTP 连接器”](#)
- [第 19.8.1 节 “关于 Internet Server API \(ISAPI\) HTTP 连接器”](#)
- [第 19.9.1 节 “关于 Netscape Server API \(NSAPI\) HTTP 连接器”](#)

mod_cluster 连接器有几个优势。

- *mod_cluster Management Protocol (MCMP)* 是应用服务器节点和 Apache Web 服务器（因此被称为“the web server”）之间的一个额外连接，应用服务器节点使用它通过自定义的 HTTP 方法集来传输服务器端的负载平衡因子及生命周期事件到 Web 服务器。
- Web 服务器的动态配置允许 JBoss EAP 6 可在运行时进行修改而无需额外的配置。
- JBoss EAP 6 执行负载平衡器因子的计算而不是依赖于 Web 服务器。这使得负载平衡度量比其他连接器更为准确。
- *mod_cluster* 提供了应用程序生命周期的细颗粒度控制。每个服务器将任何 Web 应用程序上下文生命周期事件转发给 Web 服务器，通知它启动停止对给定上下文的路由请求。这避免了最终用户看到由于不可用资源引起的 HTTP 错误。
- 您可以使用 AJP、HTTP 或 HTTPS 传输协议。

[提交 bug 报告](#)

19.5.2. 配置 mod_cluster 子系统

在管理控制台里，*mod_cluster* 选项是 Web 子系统配置区域里的一部分。点击 **Configuration** 标签页。如果你使用的是受管域，请从左上角的 **Profile** 选单里选择正确的配置集。在默认情况下，**ha** 和 **full-ha** 配置集都启用了 *mod_cluster* 子系统。如果您使用的是独立服务器，您需要使用 **standalone-ha** 或 **standalone-full-ha** 配置集来启动服务器。展开 **Web** 菜单并选择 *mod_cluster*。下表解释了这些选项。首先是总体配置，然后是会话、Web 上下文、代理、SSL 和网络的配置。每项配置在 *mod_cluster* 屏幕里都有单独的标签页。



注意

mod_cluster 配置页只对启用了 HA 群集子系统的配置集可见。这些配置集包括用于受管域的 **ha** 和 **full-ha**，或者用于独立服务器的 **standalone-ha** 和 **standalone-full-ha**。

表 19.4. *mod_cluster* 配置选项

选项	描述	CLI 命令
----	----	--------

选项	描述	CLI 命令
Load Balancing Group	如果它不为 null，请求会被发送到专门的负载均衡器上的负载均衡组。如果您不想使用负载均衡组，请将其留空。它默认是没被设置的。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=load-balancing-group,value=myGroup)</pre>
Balancer	平衡器的名称。它必须匹配 HTTPD 代理的配置。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=balancer,value=myBalancer)</pre>
Advertise Socket	用于群集广告的套接字绑定的名称。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise-socket,value=modcluster)</pre>
Advertise Security Key	包含用于广告的安全密钥的字符串。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise-security-key,value=myKey)</pre>
Advertise	是否启用广告。默认为 true 。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise,value=true)</pre>

表 19.5. mod_cluster 会话配置选项

选项	描述	CLI 命令
Sticky Session	是否对请求使用 Sticky Session 。这意味着在客户连接至专有的群集节点后，进一步的通讯将路由至相同的节点，除非这个节点不可用。它默认为 true ，也是我们推荐的设置。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session,value=true)</pre>
Sticky Session Force	如果为 true ，如果初始节点不可用，请求不会重定向到新的群集节点。相反，系统会报告失败。默认值为 false 。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session-force,value=false)</pre>
Sticky Session Remove	在失效切换时删除会话信息。默认值为 false 。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session-remove,value=false)</pre>

表 19.6. mod_cluster 的 Web Context 上下文配置选项

选项	描述	CLI 命令
Auto Enable Contexts	是否默认添加新的上下文到 mod_cluster 。默认值是 true 。如果您修改这个默认值来设置手动启用上下文，Web 应用程序将用 enable() MBean 方法、或者通过 mod_cluster （它是一个运行在虚拟主机和端口的 HTTPD 代理上的 Web 应用程序）启用它的上下文。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=auto-enable-contexts,value=true)</pre>

选项	描述	CLI 命令
Excluded Contexts	mod_cluster 应该忽略的用逗号隔开的上下文列表。如果没有指定主机，会假定主机是 localhost 。 ROOT 表示 Web 应用程序的根上下文。默认值是 ROOT, invoker, jbossws, juddi, console 。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=excluded-contexts,value="ROOT,invoker,jbossws,juddi,console")</pre>

表 19.7. mod_cluster 的代理配置选项

选项	描述	CLI 命令
Proxy URL	如果指定，MCMP 命令的 URL 里将预置它的值。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=proxy-url,value=myhost)</pre>
Proxy List	A comma-separated list of httpd proxy addresses, in the format hostname:port . This indicates the list of proxies that the mod_cluster process will attempt to communicate with initially.	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=proxy-list,value="127.0.0.1,127.0.0.2")</pre>

配置 mod_cluster 的 SSL 通讯

在默认情况下，**mod_cluster** 通讯基于未加密的 HTTP 链接上。如果您设置连接器模式为 **HTTPS**（请参考表 19.5 “**mod_cluster** 会话配置选项”），下面的设置将告诉 **mod_cluster** 在哪寻找加密连接的信息。

表 19.8. mod_cluster 的 SSL 配置选项

选项	描述	CLI 命令
----	----	--------

选项	描述	CLI 命令
ssl	是否启用 SSL。默认值为 false 。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=ssl,value=true)</pre>
Key Alias	密钥别名，当创建证书时进行选择。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=key-alias,value=jboss)</pre>
Key Store	包含客户证书的密钥库的位置	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=key-store,value=System.getProperty("user.home") + "/.keystore")</pre>
Key Store Type	密钥库的类型	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=key-store-type,value=JKS)</pre>
Key Store Provider	密钥库供应商。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=key-store-provider,value=IBMJCE)</pre>

选项	描述	CLI 命令
Password	密码，当创建证书时进行选择。	<pre> /subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/: write- attribute(name=password,value=changeit) </pre>
Trust Algorithm	信任管理工厂的算法。	<pre> /subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/: write- attribute(name=trust- algorithm,value=PKIX) </pre>
Cert File	证书文件的位置。	<pre> /subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/: write- attribute(name=certificate- file,value=\${user.home}/jboss.crt) </pre>
CRL File	证书撤销列表文件。	<pre> /subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/: write- attribute(name=crl- file,value=\${user.home}/jboss.crl) </pre>

选项	描述	CLI 命令
Max Certificate Length	信任库里的存储的证书的最大长度。默认为 5 。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=trust-max-cert-length,value=5)</pre>
Key File	证书的密钥文件的位置。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=certificate-key-file,value=\${user.home}/.keystore)</pre>
Cipher Suite	允许的加密密文套件	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=cipher-suite,value=ALL)</pre>
Certificate Encoding Algorithms	密钥管理工厂的算法。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=encoding-algorithms,value=ALL)</pre>

选项	描述	CLI 命令
Revocation URL	证书授权撤销列表的 URL。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=ca-revocation-url,value=jboss.crl)</pre>
Protocol	启用的 SSL 协议。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=protocol,value=SSLv3)</pre>

配置 mod_cluster 网络选项

可用的 **mod_cluster** 网络选项控制 **mod_cluster** 与之通讯的几个不同类型的服务的不同的超时行为。

表 19.9. mod_cluster 网络配置选项

选项	描述	CLI 命令
Node Timeout	节点的代理连接的超时时间（秒）。这是 mod_cluster 在返回错误前等待后台响应的的时间。它对应工作节点的 mod_proxy 文档里描述的超时时间。 -1 表示不会超时。请注意，在转发请求前 mod_cluster 总是使用 cping/cpong ，而 mod_cluster 使用的 connectiontimeout 值时一个 Ping 值。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=node-timeout,value=-1)</pre>
Socket Timeout	等待 HTTPD 代理到 MCMP 命令的响应并标记代理为错误状态的超时时间（毫秒）。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=socket-timeout,value=20)</pre>

选项	描述	CLI 命令
Stop Context Timeout	等待上下文（分布式上下文的待定请求完成；或非分布式上下文的活动会话销毁/过期）干净关闭的时间，它按照 stopContextTimeoutUnit 指定的单元衡量。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=stop-context-timeout,value=10)</pre>
Session Draining Strategy	<p>在卸载 Web 应用程序前是否耗尽会话。</p> <p>DEFAULT</p> <p>如果 Web 应用程序是非分布式的，在 Web 应用程序卸载以前耗尽会话。</p> <p>ALWAYS</p> <p>在 Web 应用程序（即使是非分布式 Web 应用程序）卸载以前总是耗尽会话。</p> <p>NEVER</p> <p>在 Web 应用程序（即使是非分布式 Web 应用程序）卸载以前不会耗尽会话。</p>	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=session-draining-strategy,value=DEFAULT)</pre>
Max Attempts	在放弃前，HTTP 代理试图发送给定请求到工作节点的最大次数。最小值是 1 ，表示只尝试一次。 mod_proxy 的默认值也是 1，表示不尽心重试。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=max-attempts,value=1)</pre>
Flush Packets	是否启用 Web 服务器的数据包冲刷。默认值为 false 。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=flush-packets,value=false)</pre>

选项	描述	CLI 命令
Flush Wait	在冲刷数据包到 Web 服务器前等待的时间（秒）。默认值是 -1 ， -1 表示在冲刷数据包前无限期地等待。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=flush-wait,value=-1)</pre>
Ping	等待群集节点响应的时间（秒）。默认值为 10 秒。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=ping,value=10)</pre>
SMAX	最大的 soft 空闲连接数目（和 mod_proxy 文档里的 smax 相同）。最大值取决于 httpd 线程的配置，它可以是 ThreadsPerChild 或 1 。	<pre>profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=smax,value=ThreadsPerChild)</pre>
TTL	超出 SMAX 的空闲连接可存活的时间（秒），默认为 60 。 当没有定义 nodeTimeout 时， ProxyTimeout 指令 Proxy 将被使用。如果没有定义 ProxyTimeout ，服务器的 Timeout 将被使用。其默认值是 300 秒。 nodeTimeout 、 ProxyTimeout 和 Timeout 都是在套接字级别设置的。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=ttl,value=-1)</pre>
Node Timeout	等待外部 Web 服务器的可用 Worker 进程处理请求的时间（秒）。默认值是 -1 ，表示 mod_cluster 会无限期地等待 HTTPD 工作节点处理请求。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=node-timeout,value=-1)</pre>

mod_cluster 的负载提供者配置选项

下面的 **mod_cluster** 配置选项在管理控制台里是不可用的，只能用 CLI 命令行来设置。

如果未定义动态负载处理器，简单的负载处理器将被使用。它赋予每个群集成员负载因子 **1**，并均匀地分布负载而不用考虑任何负载平衡算法。要添加它，请使用下列 CLI 命令：

```
/subsystem=modcluster/mod-cluster-config=configuration/simple-load-provider:add
```

您可以配置动态的加载提供者使用不同的算法来确定哪个群集节点接收下一个请求。您可以创建自己的加载提供者以满足环境的需要，而且您可以通过 CLI 同时添加多个活动的负载度量。默认的动态加载提供者使用 **busyness** 来确定负载度量。下面展示了动态提供者选项及可能的负载度量。

表 19.10. mod_cluster 的动态负载提供者选项

选项	描述	CLI 命令
Decay	历史度量影响力消退的因子	<pre>/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/:write-attribute(name=decay,value=2)</pre>
History	确定负载时考虑历史负载度量记录的数量。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/:write-attribute(name=history,value=9)</pre>

选项	描述	CLI 命令
Load Metric	JBoss EAP 6 的动态负载提供者附带的默认加载度量是 busyness ，它通过处理请求的线程池里的线程数量计算工作节点的负载。您可以通过和实际负载相除来设置这个度量的容量： calculated_load / capacity 。您可以在动态负载提供者里设置多个负载度量。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/load-metric=busyness/:write-attribute(name=capacity,value=1.0)</pre> <pre>/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/load-metric=busyness/:write-attribute(name=type,value=busyness)</pre> <pre>/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/load-metric=busyness/:write-attribute(name=weight,value=1)</pre>

负载度量算法

cpu

cpu 负载度量使用了平均 CPU 负载来确定哪个群集节点接受下一个工作负荷。

mem

mem 负载度量将空闲的原生内存用作负载度量。我们不鼓励使用这个度量，因为它提供的是包含缓冲和缓存的值，所以在内存管理良好的系统里它总是一个非常低的值。

heap

heap 负载度量使用 **heap** 用度来确定哪个群集接受下一个工作负荷。

sessions

session 负载度量使用活动会话的数量作为度量。

requests

requests 负载度量使用了客户请求的数量来确定哪个群集节点接受下一个工作负荷。例如，1000 表示 1000 个请求/秒会被认作“满负荷”。

send-traffic

send-traffic 负载度量使用了从工作节点发送到客户的流量。例如，默认的 512 表示如果平均的转出流量是 512KB/s 或更高时节点应该被当作满负荷状态。

receive-traffic

receive-traffic 负载度量使用了从客户发送到工作节点的流量。例如，默认的 1024 表示如果平均的转入流量是 1024KB/s 或更高时节点应该被当作满负荷状态。

busyness

这个度量代表了服务于请求的线程池的线程的数量。

例 19.1. 添加负载度量

要添加负载度量，请使用 **add-metric** 命令。

```
/subsystem=modcluster/mod-cluster-config=configuration/:add-metric(type=sessions)
```

例 19.2. 设置现有度量的值

要设置现有度量的值，请使用 **write-attribute** 命令。

```
/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/load-metric=cpu/:write-attribute(name="weight",value="3")
```

例 19.3. 修改现有度量的值

要修改现有度量的值，请使用 **write-attribute** 命令。

```
/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/load-metric=cpu/:write-attribute(name="type",value="busyness")
```

例 19.4. 删除现有度量

要删除现有的度量，请使用 **remove-metric** 命令。

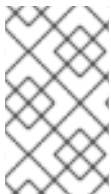
```
/subsystem=modcluster/mod-cluster-config=configuration/:remove-metric(type=sessions)
```

[提交 bug 报告](#)

19.5.3. 安装 mod_cluster 模块至 Apache HTTP 服务器或 JBoss Enterprise Web Server (ZIP 方式)

预备条件

- 要执行这个任务，您必须使用安装在 Red Hat 企业版 Linux 6 里的 Apache HTTP Server 或 JBoss Enterprise Web Server，或者作为 JBoss EAP 6 可下载组件的独立 HTTP 服务器。
- 如果你需要在 Red Hat 企业版 Linux 6 里安装 Apache HTTP 服务器，请使用《*Red Hat Enterprise Linux 6 部署指南*》里的说明。
- 如果您需要安装作为 JBoss EAP 6 的可下载组件的独立的 HTTP 服务器，请参考 [第 19.3.2 节“安装 JBoss EAP 6 附带的 Apache HTTP 服务器 \(ZIP 方式\)”](#)。
- 如果你需要安装 JBoss Enterprise Web Server，请使用《*JBoss Enterprise Web Server 安装指南*》里的说明。
- 根据操作系统和架构从 Red Hat 客户入口 <https://access.redhat.com> 下载 **Webserver Connector Natives** 软件包。这个软件包包含为操作系统预编译的 mod_cluster 的 web 服务器模块，这些模块位于 **EAP_HOME/modules/system/layers/base/native/lib/httpd/modules** 目录。
etc/ 目录包含一些配置文件示例，而 **share/** 目录则包含一些补充文档。
- 您必须用管理员权限 (root) 登录。



注意

如果您使用的是 64 位的系统，mod_cluster 的 web 服务器模块将位于：
EAP_HOME/modules/system/layers/base/native/lib64/httpd/modules。每次访问这些模块时您都必须使用这个路径。

过程 19.7. 安装 mod_cluster 模块

1. **确定您的 Apache HTTP 服务器配置的位置。**
根据您是否使用 Red Hat 企业版 Linux 的 Apache HTTP 服务器、JBoss EAP 6 是否包含作为可下载组件的独立 Apache HTTP 服务器、JBoss Enterprise Web Server 里是否有可用的 Apache HTTP 服务器，Apache HTTP 服务器配置文件的位置会有所不同。下面是这三种情况中的一种，在本节里都被称为 **HTTPD_HOME**。
 - Apache HTTP 服务器 - **/etc/httpd/**
 - JBoss EAP 6 Apache HTTP 服务器 - 根据您的系统架构需求选择的位置。
 - JBoss Enterprise Web Server 的 Apache HTTP 服务器 - **EWS_HOME/httpd/**
2. **复制这些模块到 Apache HTTP 服务器的 modules 目录。**
从解压的 Webserver Natives 归档的 **EAP_HOME/modules/system/layers/base/native/lib/httpd/modules** 目录复制四个模块（以 **.so** 结尾的文件）到 **HTTPD_HOME/modules/** 目录。
3. 对于 JBoss Enterprise Web Server，请禁用 **mod_proxy_balancer** 模块。

如果您使用 JBoss Enterprise Web Server，`mod_proxy_balancer` 默认是启用的。它和 `mod_cluster` 不兼容。要禁用它，请编辑 `HTTPD_HOME/conf/httpd.conf` 并用 `#` 注释下列加载模块的内容。下面的内容没有显示注释符号。

```
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
```

```
# LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
```

保存并关闭文件。

4. 配置 `mod_cluster` 模块。

Webserver Natives 归档包含一个 `mod_cluster.conf` 示例文件

(`EAP_HOME/modules/system/layers/base/native/etc/httpd/conf`)。这个文件可用作模版或通过复制/编辑来创建 `HTTPD_HOME/httpd/conf.d/JBoss_HTTP.conf` 文件。



注意

这个文档里的习惯是使用 `JBoss_HTTP.conf`。这个配置文件如果保存在 `conf.d/` 目录里，且扩展名为 `.conf`，它就会被加载。

在配置文件里加入下列内容：

```
LoadModule slotmem_module modules/mod_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule advertise_module modules/mod_advertise.so
```

这会让 Apache HTTP 服务器自动加载 `mod_cluster` 运行所需的模块。

5. 创建一个代理服务器 listener。

继续编辑 `HTTPD_HOME/httpd/conf.d/JBoss_HTTP.conf` 并添加下列配置，用适合您系统的值替换调大写字母表示的值。

```
Listen IP_ADDRESS:PORT
<VirtualHost IP_ADDRESS:PORT>
  <Location />
    Order deny,allow
    Deny from all
    Allow from *.MYDOMAIN.COM
  </Location>

  KeepAliveTimeout 60
  MaxKeepAliveRequests 0
  EnableMCPMReceive On

  ManagerBalancerName mycluster
  ServerAdvertise On

</VirtualHost>
```

这些指令创建了一个新的虚拟服务器，它侦听 `IP_ADDRESS:PORT`，允许来自 `MYDOMAIN.COM` 的连接，并将自己作为 `mycluster` 负载均衡器进行广告。Apache Web Server 的文档会深入讨论这些指令。要学习 `ServerAdvertise` 和 `EnableMCPMReceive` 指令以及服务器广告的含

义，请参考 [第 19.5.5 节 “为启用 mod_cluster 的 Web 服务器配置服务器的 Advertisement 属性”](#)。

保存文件并退出。

6. 重新启动 Apache HTTP 服务器。

重启 Apache HTTP 服务器的方法取决于您是否使用 Red Hat 企业版 Linux 的 Apache HTTP 服务器或包含在 JBoss Enterprise Web Server 里的 Apache HTTP 服务器。请从下面两个方法里选择一个。

- **Red Hat 企业版 Linux 6 的 Apache HTTP 服务器**

运行下列命令：

```
[root@host]# service httpd restart
```

- **JBoss Enterprise Web Server 的 Apache HTTP 服务器**

JBoss Enterprise Web Server 既可以运行在 Red Hat Enterprise Linux 也可以运行在 Microsoft Windows Server 上。两者重启 Apache HTTP 服务器的方法是不同的。

- **Red Hat Enterprise Linux**

在 Red Hat 企业版 Linux 里，JBoss Enterprise Web Server 将 Apache HTTP 服务器安装为服务。要重启 Apache HTTP 服务器，执行下面两个命令：

```
[root@host ~]# service httpd stop
[root@host ~]# service httpd start
```

- **Microsoft Windows Server**

用管理员权限执行下列命令：

```
C:\> net stop httpd
C:\> net start httpd
```

结果

Apache HTTP 服务器现在已配置为负载均衡器，且可以和运行 JBoss EAP 6 的 mod_cluster 子系统一起工作。要使 JBoss EAP 6 可以意识到 mod_cluster，请参考 [第 19.5.6 节 “配置 mod_cluster 工作节点”](#)。

[提交 bug 报告](#)

19.5.4. 安装 mod_cluster 模块至 Apache HTTP 服务器或 JBoss Enterprise Web Server (RPM 方式)

预备条件

- 要执行这个任务，您必须使用安装在 Red Hat 企业版 Linux 6 里的 Apache HTTP Server 或 JBoss Enterprise Web Server，或者作为 JBoss EAP 6 可下载组件的独立 HTTP 服务器。
- 如果你需要在 Red Hat 企业版 Linux 6 里安装 Apache HTTP 服务器，请使用《*Red Hat Enterprise Linux 6 部署指南*》里的说明。
- 如果您需要安装作为 JBoss EAP 6 的可下载组件的独立的 HTTP 服务器，请参考 [第 19.3.2 节 “安装 JBoss EAP 6 附带的 Apache HTTP 服务器 \(ZIP 方式\)”](#)。

- 如果你需要安装 JBoss Enterprise Web Server，请使用《JBoss Enterprise Web Server 安装指南》里的说明。
- 您必须用管理员权限（root）登录。
- 您必须有活动的 **jbossplatform-6-ARCH-server-VERS-rpm** RHN 频道的订阅。

RPM 安装方法对于 Red Hat 企业版 Linux 5 和 6 来说都是类似的，只是要求安装了 Apache HTTP Server 2.2.15 的 Red Hat 企业版 Linux 6 用户有少许的变化。

1. 用 YUM 安装 **mod_cluster-native** 软件包：

```
yum install mod_cluster-native
```

2. Apache HTTP Server 2.2.15:

- 如果您选择使用 Apache HTTP Server 2.2.15，您必须通过注释 **httpd.conf** 文件里的 **LoadModule proxy_balancer_module** 行来禁用 **mod_proxy_balancer** 模块。

用下列命令或手动编辑这个文件：

```
sed -i 's/^LoadModule proxy_balancer_module/#LoadModule proxy_balancer_module;/s/$//' /etc/httpd/conf/httpd.conf
```

- 如果您选择升级到 Apache HTTP Server 2.2.22，请用下列命令安装最新的版本。

```
yum install httpd
```

3. 要在引导时启动 Apache HTTP Server 服务，请执行下列命令：

- 对于 Red Hat 企业版 Linux 5 和 6：

```
service httpd add
```

- 对于 Red Hat 企业版 Linux 7：

```
systemctl enable httpd22.service
```

4. 用下列命令启动 **mod_cluster** 平衡器：

- 对于 Red Hat 企业版 Linux 5 和 6：

```
service httpd start
```

- 对于 Red Hat 企业版 Linux 7：

```
systemctl start httpd22.service
```

[提交 bug 报告](#)

19.5.5. 为启用 **mod_cluster** 的 Web 服务器配置服务器的 **Advertisement** 属性

概述

关于配置 Web 服务器和 mod_cluster 负载均衡器交互的说明，请参考 [第 19.5.3 节“安装 mod_cluster 模块至 Apache HTTP 服务器或 JBoss Enterprise Web Server \(ZIP 方式\)”](#)。需要进一步说明的配置是 *server advertisement*。

当启用 Server Advertisement 时，Web 服务器广播包含在 mod_cluster 虚拟机里指定的 IP 地址和端口的消息。关于如何配置这些值，请参考 [第 19.5.3 节“安装 mod_cluster 模块至 Apache HTTP 服务器或 JBoss Enterprise Web Server \(ZIP 方式\)”](#)。如果您的网络里 UDP 多点传送不可用，或者你更希望用静态代理服务器列表来配置工作节点，您可以禁用 Server Advertisement 并手动配置工作节点。关于配置工作节点的信息，请参考 [第 19.5.6 节“配置 mod_cluster 工作节点”](#)。

这个过程里的修改必须在和 Apache HTTP 服务器实例相关的 `httpd.conf` 里进行。对于 Red Hat 企业版 Linux，这个文件通常是 `/etc/httpd/conf/httpd.conf`，或者位于独立 Apache HTTP 服务器实例的 `etc/` 目录里。

过程 19.8. 编辑 `httpd.conf` 文件并应用这些修改。

1. 如果 `AdvertiseFrequency` 参数存在，则禁用它。

如果你的 `<VirtualHost>` 语句里有下面一行内容，请在第一个字符前用 `# (hash)` 将其注释。这个值可以不是 5。

```
AdvertiseFrequency 5
```

2. 添加指令来禁用服务器广告。

在 `<VirtualHost>` 语句里添加下列指令来禁用服务器广告。

```
ServerAdvertise Off
```

3. 禁用接收 MCPM 消息的能力。

添加下列指令到 Web 服务器来从工作节点接收 MCPM 消息。

```
EnableMCPMReceive On
```

4. 重启 Web Server。

根据你使用的是 Red Hat 企业版 Linux 还是 Microsoft Windows Server，通过下列命令之一重启 Web 服务器。

- o Red Hat Enterprise Linux

```
[root@host ]# service httpd restart
```

- o Microsoft Windows Server

```
C:\> net service http
C:\> net service httpd start
```

结果

Web 服务器不再广告 mod_cluster 代理的 IP 地址和端口。如要重新激活，您需要配置工作节点使用静态地址和端口来和代理进行通讯。详情请参考 [第 19.5.6 节“配置 mod_cluster 工作节点”](#)。

[提交 bug 报告](#)

19.5.6. 配置 mod_cluster 工作节点

概述

mod_cluster 工作节点由一个 JBoss EAP 6 服务器组成。这个服务器可以是受管域里服务器组的一部分，也可以是一个独立服务器。JBoss EAP 6 里运行的一个独立进程管理这个群集的所有节点。它被称为主节点 (Master)。关于工作节点的更多概念，请参考 第 19.1.4 节“工作节点”。关于 Web 服务器负载均衡的概述，请参考 第 19.1.3 节“HTTP 连接器概述”。

主节点只需要通过 mod_cluster 子系统配置一次。要配置 mod_cluster 子系统，请参考《管理和配置指南》里的『配置 mod_cluster 子系统』。每个工作节点都是独立配置的，所以你可以为每个要加入群集节点重复这个步骤。

如果你使用了受管域，服务器组里的每个服务器都是一个工作节点，它们共享相同的配置。因此，配置是针对整个服务器组完成的。而在独立服务器里，配置是针对单个 JBoss EAP 6 实例完成的。其余的配置步骤是相同的。

工作节点配置

- 如果你使用了独立服务器，它必须以 **standalone-ha** 配置集启动。
- 如果你使用受管域，你的服务器组必须使用 **ha** 或 **full-ha** 配置集，以及 **ha-sockets** 或 **full-ha-sockets** 套接字绑定组。JBoss EAP 6 附带满足这些要求的启用了群集的服务器组 **other-server-group**。



注意

如果使用管理 CLI 命令，它会假设你使用受管域。如果你使用的是独立服务器，请从命令行删除 **/profile=full-ha**。

过程 19.9. 配置工作节点

1. 配置网络接口。

在默认情况下，网络接口都是 **127.0.0.1**。每个容纳独立服务器或服务器组里的一个或多个服务器的物理主机的接口都需要进行配置以使用其他服务器可以看到的公共 IP 地址。

要修改 JBoss EAP 6 主机的 IP 地址，你需要关闭它并直接修改配置文件。这是因为驱动管理控制台和管理 CLI 的 Management API 依赖于稳定的管理地址。

遵循下列步骤将群集里的每个服务器的 IP 地址修改为主节点的公共 IP 地址。

- 用本节之前描述的配置集启动 JBoss EAP 服务器。
- 在 Linux 里用 **EAP_HOME/bin/jboss-cli.sh** 命令、在 Windows 服务器里使用 **EAP_HOME\bin\jboss-cli.bat** 命令启动管理 CLI。输入 **connect** 连接到 localhost 上的域控制器，或 **connect IP_ADDRESS** 连接到远程服务器上的域控制器。
- 用下列命令修改 **management**、**public** 和 **unsecure** 接口的外部 IP 地址。请确保用实际的外部 IP 地址替换命令里的 **EXTERNAL_IP_ADDRESS**。

```
/interface=management:write-attribute(name=inet-
address,value="${jboss.bind.address.management:EXTERNAL_IP_ADDRES
S}")
/interface=public:write-attribute(name=inet-
address,value="${jboss.bind.address.public:EXTERNAL_IP_ADDRESS}")
```

```
/interface=unsecure:write-attribute(name=inet-
address,value="${jboss.bind.address.unsecure:EXTERNAL_IP_ADDRESS}
"
:reload
```

您应该看到下列输出结果：

```
"outcome" => "success"
```

- d. 对于参与受管域但不是主节点的主机，你必须将主机名从 **master** 修改为其他的唯一名称。这个名称必须在从节点里是唯一的，它将被从节点用来标识群集，所以请记住这个名称。

- i. 用下列语法启动 JBoss EAP 从主机：

```
bin/domain.sh --host-config=HOST_SLAVE_XML_FILE_NAME
```

例如：

```
bin/domain.sh --host-config=host-slave01.xml
```

- ii. 启动管理 CLI。

- iii. 使用下列语法来替代主机名：

```
/host=master:write-
attribute(name="name",value=UNIQUE_HOST_SLAVE_NAME)
```

例如：

```
/host=master:write-attribute(name="name",value="host-slave01")
```

你应该看到下面的结果。

```
"outcome" => "success"
```

这修改了 **host-slave01.xml** 文件里的内容：

```
<host name="host-slave01" xmlns="urn:jboss:domain:1.6">
```

- e. 对于需要加入受管域的新配置的主机，您必须删除 **local** 元素并添加 **remote** 源及指向域控制器的 **host** 属性。这个步骤不适用于独立服务器。

- i. 用下列语法启动 JBoss EAP 从主机：

```
bin/domain.sh --host-config=HOST_SLAVE_XML_FILE_NAME
```

例如：

```
bin/domain.sh --host-config=host-slave01.xml
```

- ii. 启动管理 CLI。

- iii. 使用下列语法来指定域控制器：

```
/host=UNIQUE_HOST_SLAVE_NAME/:write-remote-domain-
controller(host=DOMAIN_CONTROLLER_IP_ADDRESS,port=${jboss.domain.master.port:9999},security-realm="ManagementRealm")
```

例如：

```
/host=host-slave01/:write-remote-domain-
controller(host="192.168.1.200",port=${jboss.domain.master.port:9999},security-realm="ManagementRealm")
```

你应该看到下面的结果。

```
"outcome" => "success"
```

这修改了 **host-slave01.xml** 文件里的内容：

```
<domain-controller>
  <remote host="192.168.1.200"
port="${jboss.domain.master.port:9999}" security-
realm="ManagementRealm"/>
</domain-controller>
```

2. 为每个从服务器配置验证。

每个从服务器都需要在域控制器或独立主服务器的 **ManagementRealm** 里创建一个用户名和密码。在域控制器或独立主服务器上，运行 **EAP_HOME/bin/add-user.sh** 命令。请用和从服务器相同的用户名添加一个用户到 **ManagementRealm**。当提示这个用户是否需要到外部的 **JBoss AS** 实例验证，请选择 **yes**。下面是这个命令的输入和输出的例子，从服务器名为 **slave1**，其密码为 **changeme**。

```
user:bin user$ ./add-user.sh

What type of user do you wish to add?
a) Management User (mgmt-users.properties)
b) Application User (application-users.properties)
(a): a

Enter the details of the new user to add.
Realm (ManagementRealm) :
Username : slave1
Password : changeme
Re-enter Password : changeme
About to add user 'slave1' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'slave1' to file '/home/user/jboss-eap-
6.0/standalone/configuration/mgmt-users.properties'
Added user 'slave1' to file '/home/user/jboss-eap-
6.0/domain/configuration/mgmt-users.properties'
Is this new user going to be used for one AS process to connect to
another AS process e.g. slave domain controller?
yes/no? yes
To represent the user add the following to the server-identities
definition <secret value="Y2hhbmdlbWU=" />
```

3. 从 **add-user.sh** 的输出里复制 Base64 编码的 **<secret>** 元素。

如果你计划验证时指定 Base64 编码的密码，请复制 `add-user.sh` 的输出里的 `<secret>` 元素值，你在下面的步骤里需要用到它。

4. 修改从主机的安全区以使用新的验证。

你可以用下列方法之一指定 `secret` 值：

- 用管理 CLI 在服务器配置文件里指定 Base64 编码的密码值。
 - a. 在 Linux 里用 `EAP_HOME/bin/jboss-cli.sh` 命令、在 Windows 服务器里使用 `EAP_HOME\bin\jboss-cli.bat` 命令启动管理 CLI。输入 `connect` 连接到 `localhost` 上的域控制器，或 `connect IP_ADDRESS` 连接到远程服务器上的域控制器。
 - b. 用下列命令指定 `secret` 值。请用前一步骤的 `add-user` 输出里返回的 `secret` 值来替换 `SECRET_VALUE`。

```
/core-service=management/security-  
realm=ManagementRealm/server-  
identity=secret:add(value="SECRET_VALUE")  
:reload
```

你应该看到下列输出结果：

```
"outcome" => "success"
```

- 配置主机通过 `vault.sh` 获取密码。

- a. 使用 `vault.sh` 脚本生成一个加密的密码。它将生成一个这样的字符串：
`VAULT::secret::password::ODVmYmJjNGMtZDU2ZC00YmNlLWE4ODMtZjQ1NWNmNDU4ZDc1TElORV9CUkVBS3ZhdWx0`。

你可以在本指南的『敏感字符串的密码阀』里找到更多信息：[第 11.13.1 节“关于保护明码文件里的敏感字符”](#)。

- b. 在 Linux 里用 `EAP_HOME/bin/jboss-cli.sh` 命令、在 Windows 服务器里使用 `EAP_HOME\bin\jboss-cli.bat` 命令启动管理 CLI。输入 `connect` 连接到 `localhost` 上的域控制器，或 `connect IP_ADDRESS` 连接到远程服务器上的域控制器。
- c. 用下列命令指定 `secret` 值。请用前一步骤生成的遮掩密码替换 `SECRET_VALUE`。

```
/core-service=management/security-  
realm=ManagementRealm/server-  
identity=secret:add(value="${VAULT::secret::password::SECRET_V  
ALUE}")  
:reload
```

你应该看到下列输出结果：

```
"outcome" => "success"
```



注意

当在阀里创建一个密码时，它必须以明文而不是 Base64 编码来指定。

- 指定密码为系统属性。

下面的例子将 `server.identity.password` 用作密码的系统属性名称。

- a. 用管理 CLI 在服务器配置文件里指定密码的系统属性。

- i. 在 Linux 里用 `EAP_HOME/bin/jboss-cli.sh` 命令、在 Windows 服务器里使用 `EAP_HOME\bin\jboss-cli.bat` 命令启动管理 CLI。输入 `connect` 连接到 `localhost` 上的域控制器，或 `connect IP_ADDRESS` 连接到远程服务器上的域控制器。
- ii. 输入下列命令来配置 `secret` 标识符以使用系统属性。

```
/core-service=management/security-  
realm=ManagementRealm/server-  
identity=secret:add(value="${server.identity.password}")  
:reload
```

您应该看到下列输出结果：

```
"outcome" => "success"
```

- b. 当你将密码指定为系统属性时，你可以用下列方法之一配置主机：

- 在命令行里以明文输入密码来启动服务器，例如：

```
-Dserver.identity.password=changeme
```



注意

密码必须以明文输入并对于任何执行 `ps -ef` 命令的用户可见。

- 将密码放在属性文件里并将文件的 URL 作为命令行参数传入。

- i. 在属性文件里添加键/值对。例如：

```
server.identity.password=changeme
```

- ii. 用命令行参数启动服务器：

```
--properties=URL_TO_PROPERTIES_FILE
```

5. 重启服务器。

从主机现在将以主机名为用户名以及加密的字符串为密码来向主服务器进行验证。

结果

你的独立服务器，或者位于受管域的服务器组里的服务器，现在已被配置为 `mod_cluster` 工作节点。如果你部署一个群集应用程序，它的会话会被复制到所有的群集节点以用于失效切换，且它可以从外部的 Web 服务器或负载均衡器接受请求。在默认情况下，群集的每个节点都可以用自动发现来发现其他节点。要配置自动发现和 `mod_cluster` 子系统的其他专有设置，请参考 [第 19.5.2 节“配置 mod_cluster 子系统”](#)。要配置 Apache HTTP 服务器，请参考 [第 19.3.5 节“将外部 Web 服务器用作 JBoss EAP 6 应用程序的 Web 前端”](#)。

[提交 bug 报告](#)

19.5.7. 在群集间移植流量

概述

在用 JBoss EAP 6 创建新的群集后，作为升级过程的一部分，您可能希望从之前的群集移植流量到新的群集里。在本节里，您会看到用于移植流量且只造成最短时间中断或停机的策略。

预备条件

- 设置的新群集 [第 19.5.2 节 “配置 mod_cluster 子系统”](#)（我们将这个群集称为 Cluster NEW）。
- 设置过剩的旧群集（我们将这个群集称为 Cluster OLD）。

过程 19.10. 群集的升级过程

1. 用预备条件里描述的步骤设置新的群集。
2. 在 Cluster NEW 和 Cluster OLD 里，请确保配置选项 **sticky-session** 设置为 **true**（默认为 **true**）。启用这个选项意味着对群集的所有新的请求都将继续发送到该节点。

```
/profile=full-ha/subsystem=modcluster/mod-cluster-
config=configuration/:write-attribute(name=sticky-
session,value=true)
```

3. 通过这里描述的过程添加 Cluster NEW 里的节点到 mod_cluster 配置里：[第 19.5.6 节 “配置 mod_cluster 工作节点”](#)。
4. 配置负载均衡器（mod_cluster）来停止群集 Cluster OLD 里的单个上下文。停止上下文（对比禁用它）将允许单个上下文优雅地关闭（最终关闭整个节点）。现有的会话将仍被服务，但不会有新的会话被指引到这些节点。被停止的上下文可能需要几分钟才能完成停止动作。

您可以用下列 CLI 命令来停止某个上下文。用和您环境相关的值替换这些参数。

```
[standalone@localhost:9999 subsystem=modcluster] :stop-
context(context=/myapp, virtualhost=default-host, waittime=50)
```

结果

您已经成功升级了 JBoss EAP 6 群集。

[提交 bug 报告](#)

19.6. APACHE MOD_JK

19.6.1. 关于 Apache mod_jk HTTP 连接器

Apache mod_jk 是一个用于兼容性目的的 HTTP 连接器。它提供了负载均衡功能，这是 JBoss Web Container 里包含的 **jboss-eap-native-webserver-connectors** 的一部分。对于所支持的平台，请参考 <https://access.redhat.com/site/articles/111663>。mod_jk 连接器是由 Apache 维护的，它的文档位于 <http://tomcat.apache.org/connectors-doc/>。

JBoss EAP 6 可以从 Apache HTTP 代理服务器接受工作负荷。代理服务器从 Web 前端接受客户请求，并将工作负荷传递给参与的 JBoss EAP 6 服务器。如果启用了 **sticky session**，相同的客户请求总会进入相同的 JBoss EAP 6 服务器，除非服务器不可用。

不像 JBoss **mod_cluster** HTTP 连接器，Apache **mod_jk** HTTP 连接器不知道服务器或服务器组上的部署的状态，所以也无法知道往哪发送工作负荷。

就像 AJP 1.3 协议之上的 **mod_cluster**、**mod_jk** 通讯。



注意

mod_cluster 是一个比 **mod_jk** 更高级的负载均衡器。**mod_cluster** 提供了 **mod_jk** 所有的功能以及其他功能。关于 **mod_cluster** 的更多信息，请参考 [第 19.5.1 节“关于 **mod_cluster** HTTP 连接器”](#)。

下一步：配置 JBoss EAP 6 实例来参与 **mod_jk** 负载均衡组。

- [第 19.3.6 节“配置 JBoss EAP 6 接受外部 Web 服务器的请求”](#)
- [第 19.6.3 节“安装 **mod_jk** 模块到 Apache HTTP 服务器（ZIP 方式）”](#)

[提交 bug 报告](#)

19.6.2. 配置 JBoss EAP 6 用 Apache Mod_jk 进行通讯

介绍

mod_jk HTTP 连接器由一个单一组件，由 Web 服务器加载的 **mod_jk.so** 模块。这个模块接收客户请求并将其转发到容器，这个例子里是 JBoss EAP 6。您也需要配置 JBoss EAP 6 来接受请求并将回复发送到 Web 服务器。

[第 19.6.3 节“安装 **mod_jk** 模块到 Apache HTTP 服务器（ZIP 方式）”](#) 里涵盖了如何配置 Apache HTTP Server。

为了 JBoss EAP 6 能够与 Apache HTTP 服务器通讯，它需要启用 AJP/1.3 连接器。在下列配置里这个连接器默认是启用的：

- 在受管域或使用 **ha** 和 **full-ha** 的服务器组里，以及 **ha** 或 **full-ha** 套接字绑定组里。默认安装里正确地配置了 **other-server-group** 服务器组。
- 在独立服务器里，**standalone-ha** 和 **standalone-full-ha** 是为群集配置的。要用这些配置集启动独立服务器，请从 **EAP_HOME/** 目录执行下列命令，并用合适的配置集进行替换。

```
[user@host bin]$ ./bin/standalone.sh --server-config=standalone-ha.xml
```

[提交 bug 报告](#)

19.6.3. 安装 **mod_jk** 模块到 Apache HTTP 服务器（ZIP 方式）

预备条件

- 要执行这个任务，您必须使用安装在受支持环境里的 Apache HTTP 服务器或安装在 JBoss Enterprise Web Server 里的 Apache HTTP 服务器。请注意，JBoss Enterprise Web Server 里安装的 Apache HTTP 服务器是 JBoss EAP 6 的一部分。
- 如果您需要安装 Apache HTTP 服务器，请使用《Red Hat 企业版 Linux 部署指南》里的说明。
- 如果你需要安装 JBoss Enterprise Web Server，请使用《JBoss Enterprise Web Server 安装指南》里的说明。
- 如果您在使用 Apache HTTP 服务器，请从 Red Hat 客户入口 <https://access.redhat.com> 下载 JBoss EAP 6 Native 组件软件包。这个软件包包含为 Red Hat 企业版 Linux 编译的 `mod_jk` 和 `mod_cluster`。如果您使用的是 JBoss Enterprise Web 服务器，它已经包含了 `mod_jk`。
- 如果您在使用 Red Hat 企业版 Linux 5 (RHEL) 和原生 Apache HTTP 服务器 (httpd 2.2.3)，请在加载 `mod_jk` 模块前加载 `mod_perl`。
- 您必须用管理员权限 (root) 登录。

过程 19.11. 安装 mod_jk 模块

1. 配置 mod_jk 模块

- 创建一个名为 `HTTPD_HOME/conf.d/mod-jk.conf` 的文件并添加下列内容：



注意

JkMount 指令指定 Apache 应该转发哪些 URL 到 `mod_jk` 模块。根据指令的配置，`mod_jk` 将接收到的 URL 转发到正确的 Servlet 容器。

要直接服务静态内容且只使用本地负载均衡器，URL 路径应该为 `/application/*`。要将 `mod_jk` 用作负载均衡器，请使用 `/*` 将所有 URL 转发给 `mod_jk`。

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile conf/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
JkLogLevel info

# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSK KEY SIZE
JkOptions +ForwardKeySize -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"
```

```
# Mount your applications
# The default setting only sends Java application data to mod_jk.
# Use the commented-out line to send all URLs through mod_jk.
# JkMount /* loadbalancer
JkMount /application/* loadbalancer

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
JkShmFile logs/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus/>
JkMount status
Order deny,allow
Deny from all
Allow from 127.0.0.1
</Location>
```

查看这些值，确保它们是合理的，然后保存文件。

b. 指定 JkMountFile 指令

除了 **mod-jk.conf** 里的 **JkMount** 指令以外，您可以指定一个包含多个转发到 **mod_jk** 的 URL 模式的文件。

- i. 添加下列内容到 **HTTPD_HOME/conf/mod-jk.conf** 文件：

```
# You can use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf/uriworkermap.properties
```

- ii. 创建一个名为 **HTTPD_HOME/conf/uriworkermap.properties** 的新文件，其中每一行对应一个 URL 模式。下面的例子展示了这个文件的语法。

```
# Simple worker configuration file
/*=loadbalancer
```

c. 复制 mod_jk.so 文件到 HTTPD 的 modules 目录



注意

只有您的 HTTP 服务器在其 **modules/** 目录下没有 **mod_jk.so** 时它才是必需的。如果您在使用 JBoss EAP 6 里包含的 Apache HTTP 服务器，您可以跳过这个步骤。

解压 Native Web Server Connectors ZIP 软件包。在 **EAP_HOME/modules/system/layers/base/native/lib/httpd/modules/** 或 **EAP_HOME/modules/system/layers/base/native/lib64/httpd/modules/** 目录里找到 **mod_jk.so** 文件（根据操作系统是 32 还是 64 位的）。

复制文件到 `HTTPD_HOME/modules/` 目录。

2. 配置 mod_jk 工作节点。

- a. 创建一个名为 `HTTPD_HOME/conf/workers.properties` 的文件。使用下面的例子作为起点，并根据需要进行修改。

```
# Define list of workers that will be used
# for mapping requests
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=node1.mydomain.com
worker.node1.type=ajp13
worker.node1.ping_mode=A
worker.node1.lbfactor=1

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
worker.node2.host=node2.mydomain.com
worker.node2.type=ajp13
worker.node2.ping_mode=A
worker.node2.lbfactor=1

# Load-balancing behavior
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2
worker.loadbalancer.sticky_session=1

# Status worker for managing load balancer
worker.status.type=status
```

关于 `workers.properties` 文件的详细描述和高级配置选项，请参考 [第 19.6.5 节“Apache Mod_jk 工作节点的配置”](#)。

3. 重启 Web Server。

重启 Web 服务器的方法取决于您是否使用 Red Hat 企业版 Linux 的 Apache HTTP 或包含在 JBoss Enterprise Web Server 里的 Apache HTTP 服务器。请从下面两个方法里选择一个。

- o **Red Hat 企业版 Linux 的 Apache HTTP 服务器**

运行下列命令：

```
[root@host]# service httpd restart
```

- o **JBoss Enterprise Web Server 的 Apache HTTP 服务器**

JBoss Enterprise Web Server 既可以运行在 Red Hat Enterprise Linux 也可以运行在 Microsoft Windows Server 上。两者重启 Web 服务器的方法是不同的。

- **Red Hat 企业版 Linux (通过 RPM 安装)**

在 Red Hat 企业版 Linux 里，JBoss Enterprise Web Server 将 Web 服务器安装为服务。要重启 Web 服务器，执行下面两个命令：

```
[root@host ~]# service httpd stop
[root@host ~]# service httpd start
```

■ Red Hat 企业版 Linux (通过 ZIP 安装)

如果您已经通过 ZIP 归档安装了 JBoss Enterprise Web Server 的 Apache HTTP 服务器，请使用 `apachectl` 命令来重启 Web 服务器。请用您解压 JBoss Enterprise Web Server 的 Apache HTTP 服务器的目录来替换 `EWS_HOME`。

```
[root@host ~]# EWS_HOME/httpd/sbin/apachectl restart
```

■ Microsoft Windows Server

用管理员权限执行下列命令：

```
C:\> net stop Apache2.2
C:\> net start Apache2.2
```

■ Solaris

请在命令行提示下用有管理权限的用户执行下列命令。请用您解压 JBoss Enterprise Web 服务器的 Apache HTTP 服务器的目录来替换 `EWS_HOME`。

```
[root@host ~] EWS_HOME/httpd/sbin/apachectl restart
```

结果

已配置 Apache HTTP 服务器使用 `mod_jk` 负载均衡器。要使 JBoss EAP 6 可以意识到 `mod_jk`，请参照第 19.3.6 节“配置 JBoss EAP 6 接受外部 Web 服务器的请求”。

[提交 bug 报告](#)

19.6.4. 安装 Mod_jk 模块至 Apache HTTP Server (RPM)

预备条件

- 要执行这个任务，您必须使用安装在受支持环境里的 Apache HTTP 服务器或安装在 JBoss Enterprise Web Server 里的 Apache HTTP 服务器。请注意，JBoss Enterprise Web Server 里安装的 Apache HTTP 服务器是 JBoss EAP 6 的一部分。
- 如果您需要安装 Apache HTTP 服务器，请使用 <https://access.redhat.com/site/documentation/> 上《Red Hat 企业版 Linux 部署指南》里的说明。
- 如果你需要安装 JBoss Enterprise Web Server，请使用《JBoss Enterprise Web Server 安装指南》里的说明。这些说明可在 <https://access.redhat.com/site/documentation/> 上找到。
- 您必须用管理员权限（root）登录。

过程 19.12. Red Hat 企业版 Linux 5 : mod_jk 和 Apache HTTP Server 2.2.3

1. 从 `jbappplatform-6-*server-5-rpm` 频道安装 `mod_jk-ap22 1.2.37` 及其依赖关系 `mod_perl`：

```
yum install mod_jk
```

2. Optional: 复制配置文件示例：

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample
/etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample
/etc/httpd/conf/workers.properties
```

您可以根据需要编辑这些文件。

3. 启动服务器：

```
service httpd start
```

注意

下列错误信息表示您的 `mod_jk` 模块在 `mod_perl` 之前被加载：

```
Cannot load /etc/httpd/modules/mod_jk.so into server:
/etc/httpd/modules/mod_jk.so: undefined symbol:
ap_get_server_description
```

要确保 `mod_perl` 模块在 `mod_jk` 之前加载，请在 `/etc/httpd/conf.d/mod_jk.conf` 里添加下列内容：

```
<IfModule !perl_module>
    LoadModule perl_module modules/mod_perl.so
</IfModule>
LoadModule jk_module modules/mod_jk.so
```

过程 19.13. Red Hat 企业版 Linux 5：mod_jk 和 JBoss EAP Apache HTTP Server 2.2.26

1. 用下列命令从 `jbappplatform-6-*-server-5-rpm` 频道安装 `mod_jk` 和最新的 Apache HTTP Server 2.2.26：

```
yum install mod_jk httpd
```

2. Optional: 复制配置文件示例：

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample
/etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample
/etc/httpd/conf/workers.properties
```

您可以根据需要编辑这些文件。

3. 启动服务器：

```
service httpd start
```

过程 19.14. Red Hat 企业版 Linux 6 : mod_jk 和 JBoss EAP Apache HTTP Server 2.2.26

1. 从 **jbappplatform-6-*server-6-rpm** 频道安装 **mod_jk-ap22 1.2.37** 和 **Apache HTTP Server 2.2.26 httpd** 软件包（任何先有的版本将被更新）：

```
yum install mod_jk httpd
```

2. **Optional:** 复制配置文件示例：

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample  
/etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample  
/etc/httpd/conf/workers.properties
```

您可以根据需要编辑这些文件。

3. 启动服务器：

```
service httpd start
```

过程 19.15. Red Hat 企业版 Linux 6 : mod_jk 和 Apache HTTP Server 2.2.15

1. 用下列命令安装 **mod_jk** 和 **Apache HTTP Server 2.2.15**：

```
yum install mod_jk
```

2. **Optional:** 复制配置文件示例：

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample  
/etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample  
/etc/httpd/conf/workers.properties
```

您可以根据需要编辑这些文件。

3. 启动服务器：

```
service httpd start
```

过程 19.16. Red Hat 企业版 Linux 7 : mod_jk 和 JBoss EAP Apache HTTP Server 2.2.26

1. 从 **jbappplatform-6-*server-6-rpm** 频道安装 **mod_jk-ap22 1.2.37** 和 **Apache HTTP Server 2.2.26 httpd** 软件包（任何先有的版本将被更新）：

```
yum install mod_jk
```

2. **Optional:** 复制配置文件示例：

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample
/etc/httpd22/conf.d/mod_jk.conf

cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample
/etc/httpd22/conf/workers.properties
```

您可以根据需要编辑这些文件。

3. 启动服务器：

```
systemctl start httpd22.service
```

[提交 bug 报告](#)

19.6.5. Apache Mod_jk 工作节点的配置

workers.properties 文件定义 mod_jk 传递客户请求的工作节点的行为。在 Red Hat 企业版 Linux 里，这个文件位于 **/etc/httpd/conf/workers.properties**。**workers.properties** 文件定义不同的 Servlet 容器的位置，以及在它们之间如何平衡负载。

这个配置分成三个部分。第一部分处理全局属性，它应用于所有的工作节点。第二部分包含应用于专有节点的设置。第三部分包含应用于专有的采用负载均衡的节点。

属性的常用结构是 **worker.WORKER_NAME.DIRECTIVE**，这里的 **WORKER_NAME** 是工作节点的唯一名称，而 **DIRECTIVE** 是应用在工作节点上的设置。

Apache Mod_jk 工作节点的配置参考

节点模版指定默认的 per-node 设置。您可以在节点设置里覆盖模版。示例模版位于 [例 19.5 “workers.properties 示例文件”](#)。

表 19.11. 全局属性

属性	描述
worker.list	mod_jk 使用的工作节点列表。这些工作节点可用来接收请求。

表 19.12. 基于每个工作节点的属性

属性	描述
type	工作节点的类型。默认类型是 ajp13 。其他可能的值还有 ajp14 、 lb 和 status 。 关于这些指令的详情，请参照 http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html 里的 Apache Tomcat Connector AJP Protocol 参考。
balance_workers	指定负载均衡器必须管理的工作节点。对于同一个负载均衡器您可以多次使用这个指令。它由一个用逗号隔开的工作节点名列表组成。它是根据每个工作节点而不是节点设置的。它影响该工作节点类型的所有节点。

属性	描述
sticky_session	指定来自相同会话的请求是否总是路由至相同的工作节点。默认值是 0 ，表示禁用 Sticky Session 。要启用 Sticky Session ，请将其设置为 1 。 Sticky Session 通常应该被启用，除非所有的请求都是 stateless 的。这是针对每个工作节点而非节点设置的。它影响该工作节点类型的所有节点。

表 19.13. 基于每个节点的属性

属性	描述
host	工作节点的主机名和 IP 地址。工作节点必须支持 ajp 协议栈。默认值是 localhost 。
port	侦听定义的协议请求的远程服务器实例的端口号码。默认值是 8009 ，这也是 AJP13 工作节点的默认侦听端口。 AJP14 工作节点的默认端口是 8011 。
ping_mode	<p>连接在哪些条件下会检测网络状态。探测仪用空的 AJP13 数据包进行 CPing，期望有 CPong 回应。使用指令标记组合来指定条件。这些标记不是用逗号或空格分隔的。ping_mode 可以是任何 C、P、I 和 A 的组合。</p> <ul style="list-style-type: none">• C - Connect。在连接至服务器后探测连接一次。用 connect_timeout 的值指定超时时间。否则使用 ping_timeout。• P - Prepost。在发送每个请求到服务器之前探测连接。用 prepost_timeout 指令指定超时时间。否则使用 ping_timeout。• I - Interval。根据 connection_ping_interval 指定的时间间隔来探测连接。否则使用 ping_timeout。• A - All。CPI 的快捷方式，它指定使用所有的连接探测仪。
ping_timeout, connect_timeout, prepost_timeout, connection_ping_interval	连接探测仪设置的超时值。这些值的单位是毫秒， ping_timeout 的默认值是 10000 。
lbfactor	为独立工作节点指定负载因子，且只应用于负载均衡器的成员工作节点。这便于为高性能的服务器赋予更多的工作负载。要赋予某个工作节点 3 倍的默认负载，将其设置为 3 ： worker.my_worker.lbfactor=3 。

例 19.5. workers.properties 示例文件

```
worker.list=node1, node2, node3

worker.balancer1.sticky_sessions=1
worker.balancer1.balance_workers=node1
worker.balancer2.sticky_session=1
worker.balancer2.balance_workers=node2,node3

worker.nodetemplate.type=ajp13
```



```

worker.nodetemplate.port=8009

worker.node1.template=nodetemplate
worker.node1.host=localhost
worker.node1.ping_mode=CI
worker.node1.connection_ping_interval=9000
worker.node1.lbfactor=1

worker.node2.template=nodetemplate
worker.node2.host=192.168.1.1
worker.node2.ping_mode=A

worker.node3.template=nodetemplate
worker.node3.host=192.168.1.2

```

Apache `mod_jk` 的进一步的配置细节超出了本文档的范畴。更多的说明请参考 <http://tomcat.apache.org/connectors-doc/> 上的 Apache 文档。

[提交 bug 报告](#)

19.7. APACHE MOD_PROXY

19.7.1. 关于 Apache `mod_proxy` HTTP 连接器

Apache 为 `httpd` 提供两个不同的代理和负载均衡模块：`mod_proxy` 和 `mod_jk`。要进一步了解 `mod_jk`，请参考 [第 19.6.1 节“关于 Apache `mod_jk` HTTP 连接器”](#)。JBoss EAP 6 支持这两种模块，但 JBoss HTTP 连接器 `mod_cluster` 和 JBoss EAP 6 及外部 HTTPD 耦合得更紧密，且是我们推荐的 `httpd` 连接器。关于所有支持的 HTTP 连接器的概述，以及其优缺点，请参考 [第 19.1.3 节“HTTP 连接器概述”](#)。

不像 `mod_jk`，`mod_proxy` 支持基于 HTTP 和 HTTPS 协议的连接。它们也支持 AJP 协议。

`mod_proxy` 可以在独立服务器或负载均衡器里配置，它 also 支持 Sticky Session 标记。

`mod_proxy` 模块要求 JBoss EAP 6 配置 HTTP、HTTPS 或 AJP web connector。这是 Web 子系统的一部分。关于配置 Web 子系统的信息，请参考 [第 17.1 节“配置 Web 子系统”](#)。

[提交 bug 报告](#)

19.7.2. 安装 `Mod_proxy` HTTP 连接器到 Apache HTTP 服务器

概述

`mod_proxy` 是 Apache 提供的一个 load-balancing 模块。这个任务代表了一个简单的配置。关于更高级的配置或其他细节，请参考 Apache 的 https://httpd.apache.org/docs/2.2/mod/mod_proxy.html 上的 `mod_proxy` 文档。关于从 JBoss EAP 6 角度来查看的 `mod_proxy` 细节，请参考 [第 19.7.1 节“关于 Apache `mod_proxy` HTTP 连接器”](#) 和 [第 19.1.3 节“HTTP 连接器概述”](#)。

预备条件

- JBoss Enterprise Web Server 的 `httpd` 或 Apache HTTP 服务器都需要先安装。独立的 Apache HTTP 服务器在 Red Hat 客户门户里是可以单独下载的。关于这个 Apache HTTP 服务器的信息，请参考 [第 19.3.2 节“安装 JBoss EAP 6 附带的 Apache HTTP 服务器（ZIP 方式）”](#)。

- **mod_proxy** 模块需被安装。Apache HTTP 服务器通常带有 **mod_proxy** 模块。对于 Red Hat 企业版 Linux，JBoss Enterprise Web Server 附带了 Apache HTTP 服务器。
- 您需要 **root** 或管理员权限来修改 Apache HTTP 服务器的配置。
- 在这些例子里，我们假设 JBoss EAP 6 配置了 HTTP 或 HTTPS Web 连接器。这是 Web 子系统配置的一部分。关于配置 Web 子系统的信息，请参考 [第 17.1 节“配置 Web 子系统”](#)。

1. 启用 HTTPD 里的 mod_proxy 模块

在 **HTTPD_HOME/conf/httpd.conf** 文件里寻找下列行。如果没有，则在文件结尾添加它们。如果存在但是以注释符号（#）开始的，请删除这些符号。保存文件。通常这些模块都会存在且已启用。

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_http_module modules/mod_proxy_http.so
# Uncomment these to proxy FTP or HTTPS
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
```

2. 添加 non-load-balancing 代理。

添加下列配置到 **HTTPD_HOME/conf/httpd.conf** 文件里，直接放在其他 **<VirtualHost>** 指令下面。请用自己的配置来替换相关的值。

这个例子使用了虚拟主机。请参考下一个步骤来使用默认的 HTTPD 配置。

```
<VirtualHost *:80>
# Your domain name
ServerName Domain_NAME_HERE

ProxyPreserveHost On

# The IP and port of JBoss EAP 6
# These represent the default values, if your httpd is on the same
host
# as your JBoss EAP 6 managed domain or server

ProxyPass / http://localhost:8080/
ProxyPassReverse / http://localhost:8080/

# The location of the HTML files, and access control information
DocumentRoot /var/www
<Directory /var/www>
Options -Indexes
Order allow,deny
Allow from all
</Directory>
</VirtualHost>
```

在进行修改后，保存这个文件。

3. 添加 load-balancing 代理。

要将 **mod_proxy** 用作负载均衡器，并将工作负载发送到多个 JBoss EAP 6 服务器，请添加下列配置到 **HTTPD_HOME/conf/httpd.conf** 文件里。这个 IP 地址是虚构的。请用自己的值来替换它。

```

<Proxy balancer://mycluster>

Order deny,allow
Allow from all

# Add each JBoss Enterprise Application Server by IP address and
port.
# If the route values are unique like this, one node will not fail
over to the other.
BalancerMember http://192.168.1.1:8080 route=node1
BalancerMember http://192.168.1.2:8180 route=node2
</Proxy>

<VirtualHost *:80>
# Your domain name
ServerName YOUR_DOMAIN_NAME

ProxyPreserveHost On
ProxyPass / balancer://mycluster/

# The location of the HTML files, and access control information
DocumentRoot /var/www
<Directory /var/www>
Options -Indexes
Order allow,deny
Allow from all
</Directory>

</VirtualHost>

```

上面的例子都使用 HTTP 协议进行通讯。如果加载合适的 `mod_proxy` 模块，您也可以使用 AJP 或 HTTPS 协议。详情请参考 http://httpd.apache.org/docs/2.2/mod/mod_proxy.html 里 Apache 的 `mod_proxy` 文档。

4. 启用会话粘连 (Sticky Session)。

Sticky sessions 表示如果客户请求最开始是转到专有的 JBoss EAP 6 节点，所有以后的请求都将发送到相同的节点，除非这个节点不可用。这通常是正确的行为。

要为 `mod_proxy` 启用 Sticky Session，请添加 `stickysession` 参数到 `ProxyPass` 语句里。这个例子也展示了一些其他的参数。更多信息请参考 Apache 的 `mod_proxy` 文档：http://httpd.apache.org/docs/2.2/mod/mod_proxy.html。

```

ProxyPass /MyApp balancer://mycluster stickysession=JSESSIONID
lbmethod=bytraffic nofailover=off

```

5. 重启 Web Server。

重启 Web 服务器，以使您的更改生效。

结果

配置 Apache HTTP 服务器使用 `mod_proxy` 以标准或负载均衡方式发送客户请求到 JBoss EAP 6 服务器或群集。要配置 JBoss EAP 6 响应这些请求，请参考第 19.3.6 节“配置 JBoss EAP 6 接受外部 Web 服务器的请求”。

[提交 bug 报告](#)

19.8. MICROSOFT ISAPI

19.8.1. 关于 Internet Server API (ISAPI) HTTP 连接器

Internet Server API (ISAPI) 是用于 Microsoft 的 Internet Information Services (IIS) Web 服务器的。您可以将 JBoss EAP 6 用作 IIS 群集里的一个工作节点。

要配置 JBoss EAP 6 参与 IIS 群集，请参考第 19.8.3 节“配置 Microsoft IIS 使用 ISAPI Redirector”。关于 ISAPI 的更多信息，请参考 [http://msdn.microsoft.com/en-us/library/ms524911\(v=VS.90\).aspx](http://msdn.microsoft.com/en-us/library/ms524911(v=VS.90).aspx)。

[提交 bug 报告](#)

19.8.2. 下载并解压用于 Microsoft IIS 的 Webserver Connector Natives

1. 在 Web 浏览器里，访问：<https://access.redhat.com>。
2. 进入 **Downloads**、**Red Hat JBoss Middleware Download Software**，然后从 **Product** 下拉列表里选择 **Enterprise Application Platform**。
3. 从 **Version** 下拉菜单里选择合适的版本。
4. 根据服务器架构选择 **Red Hat JBoss Enterprise Application Platform 6.3.0 Webserver Connector Natives for Windows Server 2008 x86_64** 或 **Red Hat JBoss Enterprise Application Platform 6.3.0 Webserver Connector Natives for Windows Server 2008 i686** 的 **Download** 选项。
5. 打开 ZIP 文件并复制 **jboss-eap-6.3/modules/system/layers/base/native/sbin** 目录的内容到服务器的某个位置。在这里我们假设这个位置是：**C:\connectors**。

[提交 bug 报告](#)

19.8.3. 配置 Microsoft IIS 使用 ISAPI Redirector

前提条件：

- 第 19.8.2 节“下载并解压用于 Microsoft IIS 的 Webserver Connector Natives”



注意

关于 Microsoft Windows Server 和 IIS 支持的配置列表，请参考 <https://access.redhat.com/site/articles/111663>。

过程 19.17. 用 IIS Manager (IIS 7) 配置 IIS Redirector

1. 点击 **Start** → **Run** 打开 IIS manager，然后输入 **inetmgr**。
2. 在左侧的树型视图面板里，展开 **IIS 7**。
3. 双击 **ISAPI and CGI Registrations** 在新窗口里打开它。
4. 在 **Actions** 面板里，点击 **Add**。Add ISAPI or CGI Restriction 窗口将会打开。

5. 指定下列值：

- **ISAPI or CGI Path:** c:\connectors\isapi_redirect.dll
- **Description:** jboss
- **Allow extension path to execute:** 选择复选框。

6. 点击 **OK** 来关闭 **Add ISAPI or CGI Restriction** 窗口。

7. 定义 **JBoss Native** 虚拟目录

- a. 右击 **Default Web Site**，然后点击 **Add Virtual Directory**。**Add Virtual Directory** 窗口将会出现。
- b. 指定下列值来添加虚拟目录：
 - **Alias:** jboss
 - **Physical Path:** C:\connectors\
- c. 点击 **OK** 保存并关闭 **Add Virtual Directory** 窗口。

8. 定义 **JBoss Native ISAPI** 重定向过滤器

- a. 在树型视图面板里，展开 **Sites → Default Web Site**。
- b. 双击 **ISAPI Filters**。**ISAPI Filters Features** 视图将会出现。
- c. 在 **Actions** 面板里，点击 **Add**。**Add ISAPI Filter** 窗口将会出现。
- d. 在 **Add ISAPI Filter** 窗口里指定下列值：
 - **Filter name:** jboss
 - **Executable:** C:\connectors\isapi_redirect.dll
- e. 点击 **OK** 保存并关闭 **Add ISAPI Filters** 窗口。

9. 启用 **ISAPI-dll** 处理程序

- a. 双击树型视图面板里的 **IIS 7** 条目。**IIS 7 Home Features View** 会被打开。
- b. 双击 **Handler Mappings**。**Handler Mappings Features View** 将会出现。
- c. 在 **IIS 7 Home Features View** 组合框里，选择 **State. Enabled and Disabled Groups** 里会出现 **Handler Mappings**。
- d. 找到 **ISAPI-dll**。如果它位于 **Disabled** 组，右击并选择 **Edit Feature Permissions**。
- e. 启用下列权限：
 - **Read**
 - **Script**

■ Execute

f. 点击 **OK** 保存并关闭 **Edit Feature Permissions** 窗口。

结果

已经配置 Microsoft IIS 使用 ISAPI Redirector。请继续阅读 [第 19.3.6 节“配置 JBoss EAP 6 接受外部 Web 服务器的请求”](#)、[第 19.8.4 节“配置 ISAPI Redirector 发送客户请求到 JBoss EAP 6”](#) 或 [第 19.8.5 节“配置 ISAPI Redirector 在多个 JBoss EAP 6 服务器间平衡客户请求”](#)。

[提交 bug 报告](#)

19.8.4. 配置 ISAPI Redirector 发送客户请求到 JBoss EAP 6

概述

本节配置了一组 JBoss EAP 6 服务器接受来自 ISAPI Redirector 的请求。它没有包括对负载平衡和高可用性失效切换的配置。如果您需要这些功能，请参考 [第 19.8.5 节“配置 ISAPI Redirector 在多个 JBoss EAP 6 服务器间平衡客户请求”](#)。

这个是在 IIS 服务器上完成的，我们假定 JBoss EAP 6 已按照 [第 19.3.6 节“配置 JBoss EAP 6 接受外部 Web 服务器的请求”](#) 进行了配置。

预备条件

- 您需要访问 IIS 服务器的完整管理员权限
- [第 19.3.6 节“配置 JBoss EAP 6 接受外部 Web 服务器的请求”](#)
- [第 19.8.3 节“配置 Microsoft IIS 使用 ISAPI Redirector”](#)

过程 19.18. 编辑属性文件和设置重定向

1. 创建一个目录来存储日志、属性文件和锁文件。

本过程剩余的部分将假定您在使用 **C:\connectors** 目录。如果您使用了不同的目录，请相应地修改这些说明。

2. 创建 **isapi_redirect.properties** 文件。

创建名为 **C:\connectors\isapi_redirect.properties** 的文件。复制下列内容到文件里。

```
# Configuration file for the ISAPI Redirector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll

# Full path to the log file for the ISAPI Redirector
log_file=c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
# Use debug only testing phase, for production switch to info
log_level=debug

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkemap.properties file
```

```
worker_mount_file=c:\connectors\uriworkermap.properties

#Full path to the rewrite.properties file
rewrite_rule_file=c:\connectors\rewrite.properties
```

如果您不想使用 **rewrite.properties** 文件，请用 # 注释最后一行。详情请参考 [步骤 5](#)。

3. 创建 uriworkermap.properties 文件

uriworkermap.properties 文件包含了部署的应用程序 URL 和处理请求的工作节点间的映射。下面的例子展示了这个文件的语法。请将您的 **uriworkermap.properties** 文件放入 **C:\connectors**。

```
# images and css files for path /status are provided by worker01
/status=worker01
/images/*=worker01
/css/*=worker01

# Path /web-console is provided by worker02
# IIS (customized) error page is used for http errors with number
greater or equal to 400
# css files are provided by worker01
/web-console/*=worker02;use_server_errors=400
/web-console/css/*=worker01

# Example of exclusion from mapping, logo.gif won't be displayed
# !/web-console/images/logo.gif=*

# Requests to /app-01 or /app-01/something will be routed to
worker01
/app-01|/*=worker01

# Requests to /app-02 or /app-02/something will be routed to
worker02
/app-02|/*=worker02
```

4. 创建 workers.properties 文件。

workers.properties 文件包含工作标签和服务器实例间的映射定义。下面的示例文件展示了这个文件的语法。请将这个文件放入 **C:\connectors** 目录。

```
# An entry that lists all the workers defined
worker.list=worker01, worker02

# Entries that define the host and port associated with these
workers

# First JBoss EAP 6 server definition, port 8009 is standard port
for AJP in EAP
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

# Second JBoss EAP 6 server definition
worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

5. 创建 `rewrite.properties` 文件。

`rewrite.properties` 文件包含了专有应用程序的简单 URL 重写规则。如下例所示，重写路径使用名称-值对来指定的。请将这个文件放入 `C:\connectors\` 目录。

```
#Simple example
# Images are accessible under abc path
/app-01/abc/=/app-01/images/
```

6. 重启 IIS 服务器。

通过 `net stop` 和 `net start` 命令重启 IIS 服务器。

```
C:\> net stop was /Y
C:\> net start w3svc
```

结果

根据应用程序配置 IIS 服务器发送客户请求到专门的 JBoss EAP 6 服务器。

[提交 bug 报告](#)

19.8.5. 配置 ISAPI Redirector 在多个 JBoss EAP 6 服务器间平衡客户请求

概述

这个配置将客户请求在多个 JBoss EAP 6 服务器间进行平衡。如果您希望将客户请求发送到专门的 JBoss EAP 6 服务器，请参考 [第 19.8.4 节“配置 ISAPI Redirector 发送客户请求到 JBoss EAP 6”](#)。

这个是在 IIS 服务器上完成的，我们假定 JBoss EAP 6 已按照 [第 19.3.6 节“配置 JBoss EAP 6 接受外部 Web 服务器的请求”](#) 进行了配置。

预备条件

- 访问 IIS 服务器的完整管理员权限。
- [第 19.3.6 节“配置 JBoss EAP 6 接受外部 Web 服务器的请求”](#)
- [第 19.8.3 节“配置 Microsoft IIS 使用 ISAPI Redirector”](#)

过程 19.19. 在多个服务器间平衡客户请求

1. 创建一个目录来存储日志、属性文件和锁文件。

本过程剩余的部分将假定您在使用 `C:\connectors\` 目录。如果您使用了不同的目录，请相应地修改这些说明。

2. 创建 `isapi_redirect.properties` 文件。

创建名为 `C:\connectors\isapi_redirect.properties` 的文件。复制下列内容到文件里。

```
# Configuration file for the ISAPI Redirector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll

# Full path to the log file for the ISAPI Redirector
```



```
log_file==c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
# Use debug only testing phase, for production switch to info
log_level=debug

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermap.properties file
worker_mount_file=c:\connectors\uriworkermap.properties

#OPTIONAL: Full path to the rewrite.properties file
rewrite_rule_file=c:\connectors\rewrite.properties
```

如果您不想使用 **rewrite.properties** 文件，请用 # 注释最后一行。详情请参考 [步骤 5](#)。

3. 创建 uriworkermap.properties 文件。

uriworkermap.properties 文件包含了部署的应用程序 URL 和处理请求的工作节点间的映射。下面的例子展示了这个文件的语法及负载均衡配置。通配符 (*) 发送不同子目录的所有请求到负载均衡器 **router**。[步骤 4](#) 里介绍了负载均衡器的配置。

将 **uriworkermap.properties** 文件放入 **C:\connectors**。

```
# images, css files, path /status and /web-console will be
# provided by nodes defined in the load-balancer called "router"
/css/*=router
/images/*=router
/status=router
/web-console|/*=router

# Example of exclusion from mapping, logo.gif won't be displayed
!/web-console/images/logo.gif=*

# Requests to /app-01 and /app-02 will be routed to nodes defined
# in the load-balancer called "router"
/app-01|/*=router
/app-02|/*=router

# mapping for management console, nodes in cluster can be enabled or
disabled here
/jkmanager|/*=status
```

4. 创建 workers.properties 文件。

workers.properties 文件包含工作标签和服务器实例间的映射定义。下面的示例文件展示了这个文件的语法。负载均衡器是在文件的结尾进行配置的，由 **worker01** 和 **worker02** 组成。**workers.properties** 文件遵循 Apache mod_jk 配置里使用的相同文件的语法。关于 **workers.properties** 文件语法的更多信息，请参考 [第 19.6.5 节“Apache Mod_jk 工作节点的配置”](#)。

将这个文件放入 **C:\connectors** 目录。

```
# The advanced router LB worker
worker.list=router,status
```

```
# First EAP server definition, port 8009 is standard port for AJP in
EAP
#
# lbfactor defines how much the worker will be used.
# The higher the number, the more requests are served
# lbfactor is useful when one machine is more powerful
# ping_mode=A - all possible probes will be used to determine that
# connections are still working

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

# Second EAP server definition
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

# Define the LB worker
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

# Define the status worker for jkmanager
worker.status.type=status
```

5. 创建 `rewrite.properties` 文件。

`rewrite.properties` 文件包含了专有应用程序的简单 URL 重写规则。如下例所示，重写路径使用名称-值对来指定的。请将这个文件放入 **C:\connectors** 目录。

```
#Simple example
# Images are accessible under abc path
/app-01/abc/=app-01/images/
```

6. 重启 IIS 服务器。

通过 **net stop** 和 **net start** 命令重启 IIS 服务器。

```
C:\> net stop was /Y
C:\> net start w3svc
```

结果

配置 IIS 服务器发送客户请求到 `workers.properties` 文件里引用的 JBoss EAP 6 服务器，在服务器间均匀地调配负载。

[提交 bug 报告](#)

19.9. ORACLE NSAPI

19.9.1. 关于 Netscape Server API (NSAPI) HTTP 连接器

Netscape Server API (NSAPI) 是一个 HTTP 连接器，它允许 JBoss EAP 6 作为 Oracle iPlanet Web Server（以前的 Netscape Web Server）里的节点。要配置这个连接器，请参考 [第 19.9.4 节“配置 NSAPI 为负载均衡群集”](#)。

[提交 bug 报告](#)

19.9.2. 在 Oracle Solaris 上配置 NSAPI Connector

概述

NSAPI connector 是一个运行在 Oracle iPlanet Web Server 内部的模块。

预备条件

- 您的服务器在 Intel 32 位、64 位、SPARC64 架构上运行 Oracle Solaris 10 或更高版本。
- 除了 NSAPI connector 以外，对于 Intel 架构，安装和配置 Oracle iPlanet Web Server 7.0.15 或更高版本；对于 SPARC 架构，7.0.14 或更高版本。
- 在每台将作为工作节点的服务器上安装和配置 JBoss EAP 6。请参考 [第 19.3.6 节“配置 JBoss EAP 6 接受外部 Web 服务器的请求”](#)。
- 从客户服务门户 <https://access.redhat.com> 下载 JBoss Native Components ZIP 软件包。

过程 19.20. 解压和设置 NSAPI Connector

1. 解压 JBoss Native 组件软件包。

本过程剩余的部分将假定 Native Components 软件包被解压至 *EAP_HOME* 目录。本过程剩余的部分，*/opt/oracle/webserver7/config/* 目录被称为 *IPLANET_CONFIG*。如果您的 Oracle iPlanet 配置目录是不同的，请相应地修改这个过程。

2. 禁用 servlet 映射。

打开 *IPLANET_CONFIG/default.web.xml* 文件并找到头部为 **Built In Server Mappings** 的部分。用注释字符 (`<!-- and -->`) 来禁用下列三个 Servlet 的映射。

- default
- invoker
- jsp

下面的配置示例展示了禁用的映射：

```
<!-- ===== Built In Servlet Mappings ===== -->
<!-- The servlet mappings for the built in servlets defined above. -
-->
<!-- The mapping for the default servlet -->
<!--servlet-mapping>
  <servlet-name>default</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping-->
<!-- The mapping for the invoker servlet -->
<!--servlet-mapping>
  <servlet-name>invoker</servlet-name>
```

```
<url-pattern>/servlet/*</url-pattern>
</servlet-mapping-->
<!-- The mapping for the JSP servlet -->
<!--servlet-mapping>
  <servlet-name>jsp</servlet-name>
  <url-pattern>*.jsp</url-pattern>
</servlet-mapping-->
```

保存文件并退出。

3. 配置 iPlanet Web Server 来加载 NSAPI connector 模块。

添加下列行到 **IPLANET_CONFIG/magnus.conf** 文件的结尾，根据您的配置修改文件路径。这些行定义了 **nsapi_redirector.so** 模块和列出工作节点和属性的 **workers.properties** 文件的位置。

```
Init fn="load-modules" funcs="jk_init,jk_service"
shlib="EAP_HOME/modules/system/layers/base/native/lib/nsapi_redirector.so" shlib_flags="(global|now)"

Init fn="jk_init"
worker_file="IPLANET_CONFIG/connectors/workers.properties"
log_level="info" log_file="IPLANET_CONFIG/connectors/nsapi.log"
shm_file="IPLANET_CONFIG/connectors/tmp/jk_shm"
```

上面的配置可用于 32 位系统。如果您使用 64 位的 Solaris，请将 **lib/nsapi_redirector.so** 修改为 **lib64/nsapi_redirector.so**。

保存文件并退出。

4. 配置 NSAPI 连接器。

您可以对 NSAPI 连接器进行基本配置，不附带负载均衡。请在配置完成前选择下列选项。

- [第 19.9.3 节 “将 NSAPI 配置为 Basic HTTP Connector”](#)
- [第 19.9.4 节 “配置 NSAPI 为负载均衡群集”](#)

[提交 bug 报告](#)

19.9.3. 将 NSAPI 配置为 Basic HTTP Connector

概述

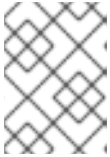
这个任务配置 NSAPI 连接器将客户请求转发到没由负载均衡或失效切换的 JBoss EAP 6 服务器。这个重定向是对每个部署（每个 URL）来进行的。关于负载均衡的配置，请参考 [第 19.9.4 节 “配置 NSAPI 为负载均衡群集”](#)。

预备条件

- 在继续当前任务前您必须完成 [第 19.9.2 节 “在 Oracle Solaris 上配置 NSAPI Connector”](#)。

过程 19.21. 设置 Basic HTTP Connector

1. 定义重定向到 JBoss EAP 6 服务器的 URL 路径。



注意

在 **IPLANET_CONFIG/obj.conf** 里，行的开头不能有空格，除非是之前行的继续。

编辑 **IPLANET_CONFIG/obj.conf** 文件。定位到以 **<Object name="default">** 开始的部分，并按下例里展示的格式添加要匹配的每个 URL 模式。**jknsapi** 字符串引用下一个步骤里将定义的 HTTP 连接器。这个例子展示了使用通配符来进行模式匹配。

```
<Object name="default">
[...]
NameTrans fn="assign-name" from="/status" name="jknsapi"
NameTrans fn="assign-name" from="/images(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/css(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/nc(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jmx-console(|/*)" name="jknsapi"
</Object>
```

2. 定义服务每个路径的工作节点。

继续编辑 **IPLANET_CONFIG/obj.conf** 文件。在刚完成编辑部分的关闭标签 (**</Object>**) 后直接添加下列内容：

```
<Object name="jknsapi">
ObjectType fn=force-type type=text/plain
Service fn="jk_service" worker="worker01" path="/status"
Service fn="jk_service" worker="worker02" path="/nc(/*)"
Service fn="jk_service" worker="worker01"
</Object>
```

上面的例子将发送到 URL 路径 **/status** 的请求重定向到名为 **worker01** 的节点，以及 **/nc/** 下的所有 URL 路径重定向到 **worker02** 工作节点。第三行指定分配给 **jknsapi** 对象的所有不匹配前面两行的 URL 都转到 **worker01**。

保存文件并退出。

3. 定义工作节点和属性。

在 **IPLANET_CONFIG/connectors/** 里创建一个名为 **workers.properties** 的文件。将下列内容粘贴到文件里，并按需要进行修改。

```
# An entry that lists all the workers defined
worker.list=worker01, worker02

# Entries that define the host and port associated with these
workers
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

workers.properties 文件使用和 **Apache mod_jk** 相同的语法。关于哪些选项可用的信息，请参考 [第 19.6.5 节 “Apache Mod_jk 工作节点的配置”](#)。

保存文件并退出。

4. 重启 iPlanet Web Server。

用下列命令来重启 iPlanet Web Server。

```
IPLANET_CONFIG/../../bin/stopserv
IPLANET_CONFIG/../../bin/startserv
```

结果

iPlanet Web Server 现在将发送到您已经配置的 URL 上的客户请求重定向到 JBoss EAP 6 上的部署。

[提交 bug 报告](#)

19.9.4. 配置 NSAPI 为负载均衡群集

介绍

这个任务配置 NSAPI 连接器将客户请求转发到负载均衡配置里的 JBoss EAP 6 服务器。要将 NSAPI 用作简单的不带负载均衡的 HTTP 连接器，请参考 [第 19.9.3 节 “将 NSAPI 配置为 Basic HTTP Connector”](#)。

必须具备的条件

- 在继续当前任务前您必须完成 [第 19.9.2 节 “在 Oracle Solaris 上配置 NSAPI Connector”](#)。

过程 19.22. 为负载均衡配置连接器

1. 定义重定向到 JBoss EAP 6 服务器的 URL 路径。



注意

在 **IPLANET_CONFIG/obj.conf** 里，行的开头不能有空格，除非是之前行的继续。

编辑 **IPLANET_CONFIG/obj.conf** 文件。定位到以 **<Object name="default">** 开始的部分，并按下例里展示的格式添加要匹配的每个 URL 模式。**jknsapi** 字符串引用下一个步骤里将定义的 HTTP 连接器。这个例子展示了使用通配符来进行模式匹配。

```
<Object name="default">
[... ]
NameTrans fn="assign-name" from="/status" name="jknsapi"
NameTrans fn="assign-name" from="/images(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/css(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/nc(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jmx-console(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jkmanager/*" name="jknsapi"
</Object>
```

2. 定义服务每个路径的工作节点。

继续编辑 **IPLANET_CONFIG/obj.conf** 文件。在之前步骤里修改的内容 (**</Object>**) 的闭合标签之后添加下列内容并按需要进行修改：

```
<Object name="jknsapi">
  ObjectType fn=force-type type=text/plain
  Service fn="jk_service" worker="status" path="/jkmanager(/*)"
  Service fn="jk_service" worker="router"
</Object>
```

这个 **jknsapi** 对象定义了用于服务映射到 **default** 对象里 **name="jknsapi"** 映射的每个路径的工作节点。除了映射 **/jkmanager/*** 的 URL 都被重定向到名为 **router** 的工作节点。

3. 定义工作节点和属性。

在 **IPLANET_CONFIG/connector/** 里创建一个名为 **workers.properties** 的文件。将下列内容粘贴到文件里，并按需要进行修改。

```
# The advanced router LB worker
# A list of each worker
worker.list=router,status

# First JBoss EAP server
# (worker node) definition.
# Port 8009 is the standard port for AJP
#

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

# Second JBoss EAP server
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

# Define the load-balancer called "router"
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

# Define the status worker
worker.status.type=status
```

workers.properties 文件使用和 **Apache mod_jk** 相同的语法。关于哪些选项可用的信息，请参考 [第 19.6.5 节“Apache Mod_jk 工作节点的配置”](#)。

保存文件并退出。

4. 重启 iPlanet Web Server。

根据您运行的是 **iPlanet Web Server 6.1** 还是 **7.0**，选择下列过程。

- **iPlanet Web Server 6.1**

```
IPLANET_CONFIG/../../stop  
IPLANET_CONFIG/../../start
```

- **iPlanet Web Server 7.0**

```
IPLANET_CONFIG/../../bin/stopserv  
IPLANET_CONFIG/../../bin/startserv
```

结果

iPlanet Web Server 将您配置的 URL 模式重定向到负载均衡配置里的 JBoss EAP 6 服务器。

[提交 bug 报告](#)

第 20 章 MESSAGING

20.1. 介绍

20.1.1. HornetQ

HornetQ 是红帽开发的多协议的异步消息系统。HornetQ 提供带有自动客户失效切换的高可用性 (HA)，保证了在服务器发生故障时消息的可靠性。HornetQ 也支持灵活的群集方案以平衡消息负载。

HornetQ 是 JBoss EAP 6 的 JMS 供应商，它被配置为 **Messaging Subsystem**。

[提交 bug 报告](#)

20.1.2. 关于 Java 消息服务 (Java Messaging Service, JMS)

消息系统允许您将多种系统用更好的可靠性松散地耦合在一起。Java Messaging Service (JMS) 供应商使用事务系统来提交或回滚原子的修改。不像基于 Remote Procedure Call (RPC) 模式的系统，消息系统主要使用异步的消息传递模式，在请求和响应间没有紧密的关系。多数消息系统也支持请求-响应模式，但这不是消息系统的主要功能。

消息系统将消息的发送者和消费者分离。发送者和消费者是完全独立的，彼此完全不知晓对方。这允许您创建灵活的、松耦合的系统。通常，大型的企业使用消息系统来实现消息总线，将多种系统松散地耦合在一起。消息总线通常组成 Enterprise Service Bus (ESB) 的核心。使用消息总线来分离不同的系统允许系统更容易成长和适应。它也提供了增加系统、减少旧系统更大的灵活性，因为它们之间的依赖性较低。

[提交 bug 报告](#)

20.1.3. 支持的消息风格

HornetQ 支持下列消息风格：

消息队列模式

消息队列模式涉及发送消息到队列里。消息位于队列后，它通常会被持久化以保证递送。一旦消息已经在队列里移动，消息系统会将其递送到某个消息消费者。进行处理后，消息消费者会确认递送。

当和点对点消息系统一起使用时，消息队列模式允许一个队列对应多个消费者，但每个消息只能被单个消费者接收。

发布-订阅模式

发布-订阅 (Publish-Subscribe) 模式允许多个发送者发送消息到服务器上的单个实体。这个实体通常被称为“主题 (Topic)”。每个主题都可被多个消费者使用，称为“订阅 (subscription)”。

每个订阅都接收发送到这个主题的每条消息的备份。这和消息队列模式不同，每条消息只能由单个消费者消费。

持久性的订阅保持发送到主题的每个消息的备份，直至订阅者消费这些消息。在服务器重启时这些备份也会保留。非持久性的订阅则和创建它们的连接持续的时间一致。

[提交 bug 报告](#)

20.2. 传输配置

20.2.1. 关于接收器（**Acceptor**）和连接器（**Connector**）

HornetQ 将连接器和接收器的概念作为消息系统的关键部分。

接收器和连接器

Acceptor

接收器（Acceptor）定义 HornetQ 服务器接受哪个类型的连接。

Connector

连接器（Connector）定义 HornetQ 客户如何使用它来连接 HornetQ 服务器。

根据配对的连接器和接收器是否处于相同的 JVM，有两种类型的连接器和接收器。

Invm 和 Netty

Invm

Invm 是 Intra Virtual Machine 的简称。当客户和服务器运行在同一个 JVM 里时可以使用它。

Netty

JBoss 项目的名称。当客户和服务器运行在不同的 JVM 里时必须使用它。

HornetQ 客户必须使用兼容其中一个服务接收器的连接器。只有 Invm 连接器可以连接 Invm 接收器，也只有 netty 连接器可以连接 netty 接收器。连接器和接收器都可以在服务器的 `standalone.xml` 和 `domain.xml` 里进行配置。您可以使用管理控制台或管理 CLI 来定义它们。

[提交 bug 报告](#)

20.2.2. 配置 Netty TCP

Netty TCP 是一个简单的、非加密的基于 TCP 套接字的传输协议。Netty TCP 可以使用旧的阻塞式 Java IO，也可以使用非阻塞式的 Java NIO。我们推荐在服务器端使用 Java NIO 以获得更好的扩充性来容纳很多并发连接。如果并发连接的数量较少，Java IO 可以获得比 NIO 更少的延迟。

我们不推荐在不信任的网络里运行 Netty TCP 连接，因为它非加密的。使用 Netty TCP 时，所有的连接都是从客户端初始化的。

例 20.1. Default EAP 配置里的 Netty TCP 配置示例

```
<connectors>
  <netty-connector name="netty" socket-binding="messaging"/>
  <netty-connector name="netty-throughput" socket-binding="messaging-
throughput">
    <param key="batch-delay" value="50"/>
  </netty-connector>
  <in-vm-connector name="in-vm" server-id="0"/>
</connectors>
<acceptors>
  <netty-acceptor name="netty" socket-binding="messaging"/>
  <netty-acceptor name="netty-throughput" socket-binding="messaging-
throughput">
    <param key="batch-delay" value="50"/>
```

```

    <param key="direct-deliver" value="false"/>
  </netty-acceptor>
  <in-vm-acceptor name="in-vm" server-id="0"/>
</acceptors>

```

配置示例也展示了 HornetQ 的 JBoss EAP 6 实现在接收器和连接器配置里是如何使用套接字绑定的。这和 HornetQ 的独立版本不同，它要求声明专门的主机和端口。

下表描述了 Netty TCP 配置属性：

表 20.1. Netty TCP 配置属性

属性	默认值	描述
batch-delay	0 毫秒	在将数据包写入到传输数据之前，HornetQ 可以在最大为 batch-delay 毫秒内批量处理写入操作。这通过增加消息传输的平均延迟，增加了很小消息的总体吞吐量。
direct-deliver	true	当消息到达服务器并被递送给等待的消费者时，在默认情况下，递送是在消息到达的相同线程里完成的。对于相对较小的消息和数量较少的消费者，这可以获得良好的延迟，但减少了吞吐量。对于更高的吞吐量，您可以将其设置为“false”。
local-address	[local address available]	对于 netty 连接器，它用来指定客户在连接远程地址时使用的本地地址。如果没有指定本地地址，那么连接器将使用任何可用的本地地址。
local-port	0	对于 netty 连接器，它用来指定客户在连接远程地址时使用哪个本地端口。如果使用 local-port default (0)，那么连接器将让系统选取一个临时端口，其有效值为 0 到 65535。
nio-remoting-threads	-1	如果使用 NIO，HornetQ 将默认使用三倍于核心（或 hyper-thread）数量（由 <code>Runtime.getRuntime().availableProcessors()</code> 确定）的线程来处理转入数据包。您可以设置自定义的值来覆盖这个值。

属性	默认值	描述
tcp-no-delay	true	如果为 true，Nagle 算法将被启用。通过减少在网络上发送的数据包时距离，这个算法有助于提高 TCP/IP 网络的效率。
tcp-send-buffer-size	32768 字节	这个参数确定了 TCP 发送缓冲的大小（字节）
tcp-receive-buffer-size	32768 字节	这个参数确定了 TCP 接收缓冲的大小（字节）
use-nio	false	如果为 true，Java 非阻塞式 NIO 将被使用。如果为 false，则使用旧的阻塞式 Java IO。如果您需要服务器处理许多并发连接，请使用非阻塞式 Java NIO，否则请使用阻塞式 Java IO。



注意

Netty TCP 属性对于所有类型的传输协议都是有效的（Netty SSL、Netty HTTP 和 Netty Servlet）。

[提交 bug 报告](#)

20.2.3. 配置 Netty Secure Sockets Layer (SSL)

Netty TCP 是一个简单的、非加密的基于 TCP 套接字的传输协议。Netty SSL 和 Netty TCP 类似，但通过 SSL 加密 TCP 连接提供了增强的安全性。下面的例子展示了单向 SSL 的 Netty 配置：



注意

下列参数大部分都既可以用于接收器也可以用于连接器。然而，一些参数只能用于接收器。参数描述解释了用于接收器和连接器间的区别。

```
<acceptors>
  <netty-acceptor name="netty" socket-binding="messaging"/>
    <param key="ssl-enabled" value="true"/>
    <param key="key-store-password" value="[keystore password]"/>
    <param key="key-store-path" value="[path to keystore file]"/>
  </netty-acceptor>
</acceptors>
```

表 20.2. Netty SSL 配置属性

属性名	默认值	描述
ssl-enabled	true	这启用了 SSL
key-store-password	[密钥库密码]	当用于接收器时，这是服务器端密钥库的密码。而用于连接器时，这是客户端密钥库的密码。如果您使用双向 SSL（也就是相互验证），它只和连接器相关。这个值可以在服务器上配置，但也可由客户端下载并使用。
key-store-path	[密钥库文件的路径]	当用于接收器时，这是保存服务器信任的所有客户的密钥的服务器端 SSL 密钥库的路径。如果您使用双向 SSL（也就是相互验证），它只和接收器相关。而用于连接器时，这是保存客户端信任的所有服务器的公共密钥的客户端 SSL 密钥库的路径。用于连接器时，它是客户端信任库的密码。这个路径可以在服务器上配置，但也可由客户端下载并使用。

如果您在配置双向 SSL 的 Netty（服务器和客户端间的相互验证），除了上面单向 SSL 例子里描述的参数外，还有三个额外的参数：

- **need-client-auth**：指定客户连接需要双向验证（相互验证）。
- **trust-store-password**：当用于接收器时，这是服务器端信任库的密码。而用于连接器时，这是客户端信任库的密码。它和单向和双向 SSL 的连接器都是相关的。这个值可以在服务器上配置，但也可由客户端下载并使用。
- **trust-store-path**：当用于接收器时，这是保存服务器信任的所有客户的密钥的服务器端 SSL 密钥库的路径。而用于连接器时，这是保存客户端信任的所有服务器的公共密钥的客户端 SSL 密钥库的路径。它和单向和双向 SSL 的连接器都是相关的。这个路径可以在服务器上配置，但也可由客户端下载并使用。

[提交 bug 报告](#)

20.2.4. 配置 Netty HTTP

Netty HTTP 通过 HTTP 协议处理数据包。在防火墙只允许 HTTP 流量通过的情况下，这很又用。Netty HTTP 使用和 Netty TCP 相同的属性已及下列额外的属性：



注意

下列参数既可以用于接收器也可以用于连接器。Netty HTTP 传输协议不允许重用标准 HTTP 端口（默认为 8080）。使用标准 HTTP 端口导致异常抛出。您可以通过标准 HTTP 端口使用 [第 20.2.5 节“配置 Netty Servlet”](#) (Netty Servlet Transport) 来处理 HornetQ 连接。

```
<socket-binding name="messaging-http" port="7080" />

<acceptors>
  <netty-acceptor name="netty" socket-binding="messaging-http">
    <param key="http-enabled" value="false"/>
    <param key="http-client-idle-time" value="500"/>
    <param key="http-client-idle-scan-period" value="500"/>
    <param key="http-response-time" value="10000"/>
    <param key="http-server-scan-period" value="5000"/>
    <param key="http-requires-session-id" value="false"/>
  </netty-acceptor>
</acceptors>
```

下表描述了配置 Netty HTTP 的其他属性：

表 20.3. Netty HTTP 配置属性

属性名	默认值	描述
http-enabled	false	true 表示启用 HTTP
http-client-idle-time	500 毫秒	在发送空的 HTTP 请求以保持连接活动之前，客户可空闲的时间。
http-client-idle-scan-period	500 毫秒	扫描空闲客户的时间间隔（毫秒）
http-response-time	10000 毫秒	在发送空的 HTTP 请求以保持连接活动之前，服务器可等待的时间。
http-server-scan-period	5000 毫秒	扫描需要响应的客户的时间间隔（毫秒）
http-requires-session-id	false	如果为 true，客户在第一次调用后将等待接收 session ID。



警告

通过 Netty HTTP 连接的客户不支持自动失效切换。

[提交 bug 报告](#)

20.2.5. 配置 Netty Servlet

Servlet 传输协议允许 HornetQ 流量通过 HTTP 传输到运行在 Servlet 引擎里的 Servlet 里，然后再重定向到 in-VM HornetQ 服务器。Netty HTTP 传输充当 Web 服务器侦听专有端口上的 HTTP 通讯。通过 servlet 传输，Servlet 引擎（可能已服务网站或其他应用程序）代理 HornetQ 流量。

为了配置 servlet 引擎使用 Netty Servlet 传输协议，您需要执行下列步骤：

- 部署 servlet：下面的例子描述了一个使用 servlet 的 Web 应用程序：

```
<web-app>
  <servlet>
    <servlet-name>HornetQServlet</servlet-name>
    <servlet-
class>org.jboss.netty.channel.socket.http.HttpTunnelingServlet</serv
let-class>
    <init-param>
      <param-name>endpoint</param-name>
      <param-value>local:org.hornetq</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>HornetQServlet</servlet-name>
    <url-pattern>/HornetQServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

初始参数 **endpoint** 指定 Servlet 将转发数据包的 Netty 接收器的 host 属性。

- 在服务器端的配置里插入 Netty servlet 接收器：下例展示了服务器配置文件里的接收器定义 (**standalone.xml** 和 **domain.xml**)：

```
<acceptors>
  <acceptor name="netty-servlet">
    <factory-class>
      org.hornetq.core.remoting.impl.netty.NettyAcceptorFactory
    </factory-class>
    <param key="use-servlet" value="true"/>
    <param key="host" value="org.hornetq"/>
  </acceptor>
</acceptors>
```

- 最后一步是在服务器配置文件 (**standalone.xml** 和 **domain.xml**) 里为客户定义接收器：

```
<netty-connector name="netty-servlet" socket-binding="http">
  <param key="use-servlet" value="true"/>
  <param key="servlet-path" value="/messaging/HornetQServlet"/>
</netty-connector>
```

- 添加下列接收器配置，您也可以使用基于 SSL 的 Servlet 传输：

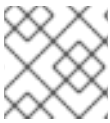
```
<netty-connector name="netty-servlet" socket-binding="https">
```

```
<param key="use-servlet" value="true"/>
<param key="servlet-path" value="/messaging/HornetQServlet"/>
<param key="ssl-enabled" value="true"/>
<param key="key-store-path" value="path to a key-store"/>
<param key="key-store-password" value="key-store password"/>
</connector>
```



警告

通过 HTTP tunneling servlet 连接的客户端不支持自动失效切换。



注意

Netty servlet 无法用于配置 EAP 服务器以设立 HornetQ 群集。

[提交 bug 报告](#)

20.3. 关于 JAVA 命名和目录接口（JAVA NAMING AND DIRECTORY INTERFACE , JNDI）

Java Naming and Directory Interface (JNDI) 是一个用于命名和目录服务的标准的 Java API。它允许用基于 Java 的技术来发现和组织分布式计算环境里的命名组件。

[提交 bug 报告](#)

20.4. 处理大型消息

20.4.1. 处理大型消息

即使客户端或服务器的内存有限，HornetQ 也支持大型消息。大型消息可以流化（streamed）或进行压缩以更高效地传输。用户可以设置消息主体里的 **InputStream** 以发送发想消息。当消息被发送时，HornetQ 读取这个 **InputStream** 并以片段形式将数据传输到服务器。

客户或服务器从不会将大型消息的主体全部存储在内存里。消费首先会接收到具有空主体大型消息，然后设置消息上的 **OutputStream** 来将数据流分割成片段并存储在磁盘文件里。

[提交 bug 报告](#)

20.4.2. 配置 HornetQ 大型消息

配置服务器

在独立服务器模式下，大型消息存储在 **EAP_HOME/standalone/data/largemessages** 目录里。在域模式下，大型消息存储在 **EAP_HOME/domain/servers/SERVERNAME/data/largemessages** 目录里。配置属性 **large-messages-directory** 指定大型消息存储的位置。



重要

为了达到最好的性能，我们推荐将大型消息直接存储在和消息日志不同的物理位置或 `paging` 目录里。

[提交 bug 报告](#)

20.4.3. 配置参数

您可以通过设置不同的参数来配置 HornetQ 大型消息：

在客户端使用 HornetQ Core API

如果在客户端使用 HornetQ Core API，您需要设置 `ServerLocator.setMinLargeMessageSize` 参数来指定大型消息的最小尺寸。大型消息的最小尺寸 (`min-large-message-size`) 默认为 100KB。

```

ServerLocator locator = HornetQClient.createServerLocatorWithoutHA(new
    TransportConfiguration(NettyConnectorFactory.class.getName()))

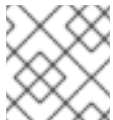
locator.setMinLargeMessageSize(25 * 1024);

ClientSessionFactory factory =
    HornetQClient.createClientSessionFactory();

```

为 Java Messaging Service (JMS) 客户配置服务器

如果您使用 Java Messaging Service (JMS)，您需要通过服务器配置文件 (`standalone.xml` 和 `domain.xml`) 里的 `min-large-message-size` 属性指定大型消息的最小尺寸。大型消息的最小尺寸 (`min-large-message-size`) 默认为 100KB。



注意

`min-large-message-size` 属性的单位应该为字节。

您可以选择压缩大型消息以获得快速和高效的传输。所有的压缩/解压操作都是在客户端来处理的。如果压缩的消息比 `min-large-message-size` 要小，它将作为常规消息发送给服务器。如果使用 Java Messaging Service (JMS)，您可以设置服务器定位器或连接工厂上的布尔型属性 `compress-large-messages` 为 "true" 来压缩大型消息。

```

<connection-factory name="ConnectionFactory">
    <connectors>
        <connector-ref connector-name="netty"/>
    </connectors>
    ...
    <min-large-message-size>204800</min-large-message-size>
    <compress-large-messages>true</compress-large-messages>
</connection-factory>

```

[提交 bug 报告](#)

20.5. 分页

20.5.1. 关于分页

HornetQ 支持多个消息队列，每个队列可包含数百万条消息。HornetQ 服务器只有有限的内存，因此无法同时在内存里存储所有的消息队列。

分页是HornetQ 服务器使用的一个机制，它透明地将内存里的消息分页以在有限的内存里容纳大型消息队列。

当某个地址的内存里的消息大小超过了指定的值，HornetQ 会开始将消息分页。



注意

HornetQ 分页默认是启用的。

[提交 bug 报告](#)

20.5.2. 分页文件

在将消息存储在多个文件的文件系统里，每个地址都对应一个单独的文件夹。这些保存消息的文件被称为分页文件（Page file）。每个文件包含最多为指定大小（**page-size-bytes**）的消息。

系统按需要浏览分页文件，只要分页里的所有消息都被客户接收，它会尽快删除分页文件。

带有选择器（Selector）的消费者也将浏览分页文件，但它会忽略不符合条件的消息。

[提交 bug 报告](#)

20.5.3. 配置分页文件夹

全局分页参数是在服务器配置文件（`standalone.xml` 和 `domain.xml`）里指定的。您可以用 ***paging-directory*** 参数配置分页目录/文件夹的位置：

```
<hornetq-server>
...
<paging-directory>/location/paging-directory</paging-directory>
...
</hornetq-server>
```

paging-directory 参数用来指定存储分页文件的位置/文件夹。HornetQ 为这个分页目录里的每个分页地址创建一个文件夹。分页文件存储在这些文件夹里。

默认的分页目录是 **EAP_HOME/standalone/data/messagingpaging**（独立模式）和 **EAP_HOME/domain/servers/SERVERNAME/data/messagingpaging**（域模式）。

[提交 bug 报告](#)

20.5.4. 分页模式

当递送到某个地址的消息超过了指定的大小，该地址将进入分页模式（**page/paging mode**）。



注意

分页是针对单个地址的。如果您为某个地址配置了 *max-size-bytes*，就表示每个匹配的
地址将有这个最大尺寸的限制。然而，这并不意味着所有匹配的地址的总共大小被限制为
max-size-bytes。

您可以在服务器配置文件（*standalone.xml* 和 *domain.xml*）里为某个地址配置最大字节数（*max-size-bytes*）。

```
<address-settings>
  <address-setting match="jms.someaddress">
    <max-size-bytes>104857600</max-size-bytes>
    <page-size-bytes>10485760</page-size-bytes>
    <address-full-policy>PAGE</address-full-policy>
  </address-setting>
</address-settings>
```

下表描述了地址设置的参数：

表 20.4. 分页地址设置

元素	默认值	描述
max-size-bytes	10485760	它用来指定进入分页模式前这个地址可以容纳的最大内存大小。
page-size-bytes	2097152	它用来指定分页系统上每个分页文件的大小。
address-full-policy	PAGE	这个属性的值用于分页决策。您可以设置为这些值： PAGE ：启用分页且将超过限额的消息转储在磁盘里， DROP ：丢弃超过限制的消息， FAIL ：丢弃消息并发送异常到客户消息生产者， BLOCK ：当发送的消息超过限额时阻塞客户消息生产者。
page-max-cache-size	5	系统将保持内存里的分页文件为 <i>page-max-cache-size</i> 以优化分页导航过程中的输入/输出。



重要

如果当到达最大大小时您不想将消息分页，您可以选择设置某个地址的 *address-full-policy* 为 **DROP**、**FAIL** 和 **BLOCK** 分别来简单地丢弃消息、丢弃消息并在客户端抛出异常、或者阻止生产者发送更多的消息。在默认配置里，所有的地址都在达到 *max-size-bytes* 后开始对消息分页。

具有多个队列的地址

当消息被路由至某个绑定多个队列的地址时，这个消息在内存里只有一个备份。每个队列都只处理原始消息备份的引用。因此，只有当引用原始消息的所有队列已经递送消息后，内存才会被释放。



注意

单个的 **lazy** 队列/订阅可能降低整个地址的输入/输出性能，这是因为所有的队列都将消息通过分页系统上的额外存储进行发送。

[提交 bug 报告](#)

20.6. 配置

20.6.1. 配置 JMS 服务器

要为 HornetQ 配置 JMS 服务器，请编辑服务器配置文件。对于域服务器，服务器配置包含在 **EAP_HOME/domain/configuration/domain.xml** 里；而对于独立服务器，则位于 **EAP_HOME/standalone/configuration/standalone-full.xml** 文件里。

服务器配置文件里的 `<subsystem xmlns="urn:jboss:domain:messaging:1.4">` 元素包含了所有的 JMS 配置。按需要为 JNDI 添加任何 **JMS ConnectionFactory**、**Queue** 或 **Topic** 实例。

1. 启用 JBoss EAP 6 里的 JMS 子系统。

在 `<extensions>` 元素里，检查是否有下面一行且没被注释：

```
<extension module="org.jboss.as.messaging"/>
```

2. 添加基本的 JMS 子系统。

如果 Messaging 子系统没有出现在你的配置文件里，请添加它。

- a. 查找对应你所使用的配置集的 `<profile>`，并定位它的 `<subsystems>` 标签。
- b. 将下列 XML 内容复制到 `<profile>` 标签下。

```
<subsystem xmlns="urn:jboss:domain:messaging:1.4">
  <hornetq-server>
    <!-- ALL XML CONFIGURATION IS ADDED HERE -->
  </hornetq-server>
</subsystem>
```

所有进一步的配置都将添加到空行的上方。

3. 添加基本的 JMS 配置。

在 `<subsystem xmlns="urn:jboss:domain:messaging:1.4"><hornetq-server>` 标签下的空行里添加下列 XML 内容：

```
<journal-min-files>2</journal-min-files>
<journal-type>NIO</journal-type>
<persistence-enabled>true</persistence-enabled>
```

定制上面的值以满足你的需要。



警告

journal-file-size 的值必须大于或等于 **min-large-message-size** (默认值为 100KiB)，否则服务器无法存储这个消息。

4. 在 HornetQ 里添加连接工厂实例

客户使用 **JMS ConnectionFactory** 对象来创建到服务器的连接。要添加 JMS 连接工厂对象到 HornetQ 里，请为每个连接工厂包含一个 **<jms-connection-factories>** 标签和 **<connection-factory>** 元素：

```
<jms-connection-factories>
  <connection-factory name="InVmConnectionFactory">
    <connectors>
      <connector-ref connector-name="in-vm"/>
    </connectors>
    <entries>
      <entry name="java:/ConnectionFactory"/>
    </entries>
  </connection-factory>
  <connection-factory name="RemoteConnectionFactory">
    <connectors>
      <connector-ref connector-name="netty"/>
    </connectors>
    <entries>
      <entry
name="java:jboss/exported/jms/RemoteConnectionFactory"/>
    </entries>
  </connection-factory>
  <pooled-connection-factory name="hornetq-ra">
    <transaction mode="xa"/>
    <connectors>
      <connector-ref connector-name="in-vm"/>
    </connectors>
    <entries>
      <entry name="java:/JmsXA"/>
    </entries>
  </pooled-connection-factory>
</jms-connection-factories>
```

5. 配置 netty 连接器和接收器

这个 JMS 连接工厂使用 **netty** 接收器和连接器。它们是部署在服务器配置文件里的连接器和接收器对象的引用。连接器对象定义用来连接 HornetQ 服务器的传输和参数。接收器对象则确定 HornetQ 服务器接受的连接的类型。

要配置 **netty** 连接器，请包含下列设置：

```
<connectors>
  <netty-connector name="netty" socket-binding="messaging"/>
  <netty-connector name="netty-throughput" socket-
```

```

binding="messaging-throughput">
  <param key="batch-delay" value="50"/>
</netty-connector>
<in-vm-connector name="in-vm" server-id="0"/>
</connectors>

```

要配置 **netty** 连接器，请包含下列设置：

```

<acceptors>
  <netty-acceptor name="netty" socket-binding="messaging"/>
  <netty-acceptor name="netty-throughput" socket-
binding="messaging-throughput">
    <param key="batch-delay" value="50"/>
    <param key="direct-deliver" value="false"/>
  </netty-acceptor>
  <in-vm-acceptor name="in-vm" server-id="0"/>
</acceptors>

```

6. 复查配置。

如果您遵照了之前的步骤，您的消息子系统应该类似于：

```

<subsystem xmlns="urn:jboss:domain:messaging:1.4">
  <hornetq-server>
    <journal-min-files>2</journal-min-files>
    <journal-type>NIO</journal-type>
    <persistence-enabled>true</persistence-enabled>
    <jms-connection-factories>
      <connection-factory name="InVmConnectionFactory">
        <connectors>
          <connector-ref connector-name="in-vm"/>
        </connectors>
        <entries>
          <entry name="java:/ConnectionFactory"/>
        </entries>
      </connection-factory>
      <connection-factory name="RemoteConnectionFactory">
        <connectors>
          <connector-ref connector-name="netty"/>
        </connectors>
        <entries>
          <entry
name="java:jboss/exported/jms/RemoteConnectionFactory"/>
        </entries>
      </connection-factory>
      <pooled-connection-factory name="hornetq-ra">
        <transaction mode="xa"/>
        <connectors>
          <connector-ref connector-name="in-vm"/>
        </connectors>
        <entries>
          <entry name="java:/JmsXA"/>
        </entries>
      </pooled-connection-factory>
    </jms-connection-factories>
  </hornetq-server>
</subsystem>

```

```

</jms-connection-factories>
<connectors>
  <netty-connector name="netty" socket-
binding="messaging"/>
  <netty-connector name="netty-throughput" socket-
binding="messaging-throughput">
    <param key="batch-delay" value="50"/>
  </netty-connector>
  <in-vm-connector name="in-vm" server-id="0"/>
</connectors>
<acceptors>
  <netty-acceptor name="netty" socket-
binding="messaging"/>
  <netty-acceptor name="netty-throughput" socket-
binding="messaging-throughput">
    <param key="batch-delay" value="50"/>
    <param key="direct-deliver" value="false"/>
  </netty-acceptor>
  <in-vm-acceptor name="in-vm" server-id="0"/>
</acceptors>
</hornetq-server>
</subsystem>

```

7. 配置套接字绑定组

netty 连接器引用了 **messaging** 和 **messaging-throughput** 套接字绑定。**messaging** 套接字使用了端口 5445，而 **messaging-throughput** 套接字绑定使用了端口 5455。**<socket-binding-group>** 标签位于服务器配置文件的单独部分。请确保下列套接字绑定出现在 **<socket-binding-groups>** 元素里：

```

<socket-binding-group name="standard-sockets" default-
interface="public" port-offset="{${jboss.socket.binding.port-
offset:0}}">
  ...
  <socket-binding name="messaging" port="5445"/>
  <socket-binding name="messaging-throughput" port="5455"/>
  ...
</socket-binding-group>

```

8. 可选：添加 queue 实例到 HornetQ 里

有四种方法可设置 HornetQ 的队列实例（或 JMS 目的地）。

○ 使用管理控制台

要使用管理控制台，服务器必须已经以 **Message-Enabled** 模式启动。您可以用 **-c** 选项并强制使用 **standalone-full.xml**（对于独立服务器）配置文件来做到这一点。例如，在独立模式下，下列命令将以 **message-enabled** 模式启动服务器：

```
./standalone.sh -c standalone-full.xml
```

服务器启动后，请登录管理控制台并选择 **Configuration** 标签页。展开 **Subsystems** 菜单，然后展开 **Messaging** 菜单并点击 **Destinations**。在 JMS Messaging Provider 表的 **Default** 的右侧点击 **View**，然后点击 **Add** 输入 JMS 目的地的细节。

- 使用管理 CLI :

首先, 连接至管理 CLI :

```
bin/jboss-cli.sh --connect
```

然后, 进入消息子系统 :

```
cd /subsystem=messaging/hornetq-server=default
```

最后, 执行 **add** 操作, 用自己的内容替换下例里的值 :

```
./jms-queue=testQueue:add(durable=false,entries=
["java:jboss/exported/jms/queue/test"])
```

- 创建一个 JMS 配置文件并将其添加至 **deployments** 文件夹

从创建 JMS 配置文件 *example-jms.xml* 开始。添加下列条目, 用自己的设置替换相关的值 :

```
<?xml version="1.0" encoding="UTF-8"?>           <messaging-
deployment xmlns="urn:jboss:messaging-deployment:1.0">
  <hornetq-server>
    <jms-destinations>
      <jms-queue name="testQueue">
        <entry name="queue/test"/>
        <entry
name="java:jboss/exported/jms/queue/test"/>
      </jms-queue>
      <jms-topic name="testTopic">
        <entry name="topic/test"/>
        <entry
name="java:jboss/exported/jms/topic/test"/>
      </jms-topic>
    </jms-destinations>
  </hornetq-server>
</messaging-deployment>
```

保存文件到 **deployments** 文件夹并进行部署。

- 在 JBoss EAP 6 配置文件里添加条目。

以 *standalone-full.xml* 为例子, 找到里面的消息子系统。

```
<subsystem xmlns="urn:jboss:domain:messaging:1.4">
```

添加下列条目, 并用自己的设置替换相关的值。您需要在 **</jms-connection-factories>** 标签后但在 **</hornetq-server>** 元素前添加这些内容 :

```
<jms-destinations>
  <jms-queue name="testQueue">
    <entry name="queue/test"/>
    <entry name="java:jboss/exported/jms/queue/test"/>
  </jms-queue>
  <jms-topic name="testTopic">
```



```
        <entry name="topic/test"/>
        <entry name="java:jboss/exported/jms/topic/test"/>
    </jms-topic>
</jms-destinations>
```

9. 执行其他的配置

如果您需要其他的设置，请复查 *EAP_HOME/docs/schema/jboss-as-messaging_1_4.xsd* 里的 DTD。

[提交 bug 报告](#)

20.6.2. 配置 JMS 地址设置

JMS 子系统有几个可配置选项，它们控制消息如何和何时递送、可以多少次重试、何时过期。这些配置选项都存在于 **<address-settings>** 配置元素里。

地址配置的一个常见功能是映射多个地址的语法，也称作通配符。

通配符语法

地址通配符可用来映射多个类似的地址到单个语句，就像多个系统使用星号（*）字符来映射多个文件或字符串。下面的字符在通配符语句里都有特定的含义。

表 20.5. JMS 通配符语法

字符	描述
.(单个句号)	表示通配符表达式里单词间的空格。
# (井字键)	匹配零或多个单词的任意序列。
* (星号)	匹配单个单词。

表 20.6. JMS 通配符示例

实例	描述
news.europe.#	匹配 news.europe 、 news.europe.sport 、 news.europe.politic ，但不匹配 news.usa 或 europe 。
news.*	匹配 news.europe 但不匹配 news.europe.sport 。
news.*.sport	匹配 news.europe.sport 和 news.usa.sport 但不匹配 news.europe.politics 。

例 20.2. 默认的地址设置配置

这个例子里的值适用于本节的剩余内容。

```
<address-settings>
  <!--default for catch all-->
  <address-setting match="#">
    <dead-letter-address>jms.queue.DLQ</dead-letter-address>
    <expiry-address>jms.queue.ExpiryQueue</expiry-address>
    <redelivery-delay>0</redelivery-delay>
    <max-size-bytes>10485760</max-size-bytes>
    <address-full-policy>BLOCK</address-full-policy>
    <message-counter-history-day-limit>10</message-counter-history-
day-limit>
  </address-setting>
</address-settings>
```

表 20.7. JMS 地址设置的描述

元素	描述	默认值	类型
address-full-policy	确定当已指定 max-size-bytes 的地址已满时发生的事情。	PAGE	STRING
dead-letter-address	如果指定了 Dead Letter 地址，如果 max-delivery-attempts 次递送尝试已失败，消息会移动到这个地址。否则，未递送的消息将被舍弃。这里允许使用通配符。	jms.queue.DLQ	STRING
expiry-address	如果指定了过期地址，过期的消息将被送到这个地址或匹配的地址，而不会被舍弃。这里允许使用通配符。	jms.queue.ExpiryQueue	STRING
last-value-queue	定义队列是否只使用最后的值。	false	BOOLEAN
max-delivery-attempts	在消息被发送到 dead-letter-address 或舍弃前重新递送的最多次数。	10	INT
max-size-bytes	最大值（字节）。	10485760L	LONG
message-counter-history-day-limit	消息计数器历史的每日限制。	10	INT
page-max-cache-size	在页面导航过程中保持在内存里以优化 IO 的页面文件的数量。	5	INT

元素	描述	默认值	类型
page-size-bytes	页面大小。	5	INT
redelivery-delay	在重新递送消息间延迟的时间，单位为毫秒。如果为 0 ，重递送将无限期进行下去。	0L	LONG
redistribution-delay	当队列上最后一个消费者关闭时重新发布任何消息前等待的时间。	-1L	LONG
send-to-dla-on-no-route	设置消息不路由至任何队列而发送至 Dead Letter 地址 (DLA) 的条件的参数。	false	BOOLEAN

- **配置地址设置和模式属性**

选择管理 CLI 或管理控制台来配置您的模式属性。

- **用管理 CLI 配置地址设置**

使用管理 CLI 配置地址设置。

- a. **添加新的模式**

如果有需要，请使用 **add** 操作来创建新的地址设置。您可以在管理 CLI 会话的根目录里运行这个命令，下面的例子创建了一个名为 *patternname* 的模式，它的 **max-delivery-attempts** 属性声明为 5。下面展示了对独立服务器和受管域的 **full** 配置集的编辑。

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default/address-setting=patternname/:add(max-delivery-attempts=5)
```

```
[domain@localhost:9999 /] /profile=full/subsystem=messaging/hornetq-server=default/address-setting=patternname/:add(max-delivery-attempts=5)
```

- b. **编辑模式属性**

使用 **write** 操作来编写新的属性的值。您可以使用 **tab completion** 来帮助输入并提示所有可用的值。下面的例子将 **max-delivery-attempts** 的值更新为 10。

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default/address-setting=patternname/:write-attribute(name=max-delivery-attempts,value=10)
```

```
[domain@localhost:9999 /] /profile=full/subsystem=messaging/hornetq-server=default/address-setting=patternname/:write-attribute(name=max-delivery-attempts,value=10)
```

c. 确认模式属性

通过 `include-runtime=true` 参数运行 `read-resource` 操作来确认值已修改以开放服务器模型里所有当前的值。

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-
server=default/address-setting=patternname/:read-resource
```

```
[domain@localhost:9999 /]
/profile=full/subsystem=messaging/hornetq-
server=default/address-setting=patternname/:read-resource
```

o. 用管理控制台配置地址设置

使用管理控制台配置地址设置。

- 登录至受管域或独立服务器的管理控制台。
- 选择屏幕顶部的 **Configuration** 标签页。对于域模式，在屏幕左上角的 **Profile** 菜单里选择配置集。只有 **full** 和 **full-ha** 配置集启用了 **messaging** 子系统。
- 展开 **Messaging** 菜单，然后选择 **Destinations**。
- JMS 供应商列表将会出现。在 **default** 配置里只有一个供应商，就是 **default**。点击 **View** 链接查看这个供应商的详细设置。
- 点击 **Addressing Settings** 标签页。点击 **Add** 按钮添加新模式，或者选择已有模式并点击 **Edit** 按钮更新设置。
- 如果您在添加新的模式，**Pattern** 字段引用 **address-setting** 元素的 **match** 参数。您也可以编辑 **Dead Letter Address**, **Expiry Address**, **Redelivery Delay** 和 **Max Delivery Attempts**。其他选项需要用管理 CLI 进行配置。

[提交 bug 报告](#)

20.6.3. 用 HornetQ 配置消息系统

我们推荐在 JBoss EAP 6 里配置消息系统的方法是通过管理控制台或管理 CLI。您可以用这些工具来进行持久性的修改而无需手动编辑 `standalone.xml` 或 `domain.xml` 文件。但熟悉默认配置文件的消息组件是有用处的，而使用管理工具的示例的文档里包含了配置文件的片段以供参考。

[提交 bug 报告](#)

20.6.4. 为 HornetQ 启用日志

您可以用下列途径在 EAP 6.x 里为 HornetQ 启用日志：

- 手动编辑服务器配置文件 (`standalone-full.xml` 和 `standalone-full-ha.xml`)
- 用 CLI 编辑服务器配置文件

过程 20.1. 手动编辑服务器配置文件来设置 HornetQ 日志

1. 打开服务器配置文件进行编辑。例如：`standalone-full.xml` 和 `standalone-full-ha.xml`

2. 进入文件里的日志子系统配置部分。默认的配置类似于：

```
<logger category="com.arjuna">
  <level name="TRACE"/>
</logger>
...
<logger category="org.apache.tomcat.util.modeler">
  <level name="WARN"/>
</logger>
....
```

3. 添加下例所示的 **org.hornetq** logger 类别及想要的日志级别：

```
<logger category="com.arjuna">
  <level name="TRACE"/>
</logger>
...
<logger category="org.hornetq">
  <level name="INFO"/>
</logger>
....
```

结果

HornetQ 日志已启用且根据配置的日志级别处理日志信息。

用 CLI 编辑服务器配置文件来设置 HornetQ 日志

您也可以使用 CLI 来添加 **org.hornetq** logger 类别及想要的日志级别到服务器配置文件里。更多信息请参考 第 14.3.2 节 “在 CLI 里配置日志类别”。

[提交 bug 报告](#)

20.6.5. 配置 HornetQ Core Bridge

例 20.3. HornetQ Core Bridge 的配置示例：

这个例子中的值将用于本节里剩余的内容。

```
<bridges>
  <bridge name="myBridge">
    <queue-name>jms.queue.InQueue</queue-name>
    <forwarding-address>jms.queue.OutQueue</forwarding-address>
  <ha>true</ha>
    <reconnect-attempts>-1</reconnect-attempts>
    <use-duplicate-detection>true</use-duplicate-detection>
    <static-connectors>
      <connector-ref>
        bridge-connector
      </connector-ref>
    </static-connectors>
  </bridge>
</bridges>
```

表 20.8. HornetQ Core Bridge 属性

属性	描述
name	服务器上所有的桥都必须有一个唯一名称。
queue-name	这个强制性的参数是桥消费的本地队列的唯一名称。在启动时实例化桥时这个队列必须已经存在。
forwarding-address	这是消息将转发至的目标服务器上的地址。如果没有指定转发地址，那么消息的原始地址将被保留。
ha	这个可选参数指定这个桥是否应该支持高可用性。 true 表示它将连接至群集里的任何可用的服务器并支持失效切换。默认值是 false 。
reconnect-attempts	这个可选参数指定桥在放弃和关闭前应该尝试重新连接的总共次数。 -1 表示无限次重试。默认值是 -1 。
use-duplicate-detection	这个可选参数指定桥是否自动插入重复的 ID 属性至它转发的每条消息里。
static-connectors	static-connectors 是指向其他地方定义的连接元素 connector-ref 元素的列表。连接器封装使用的传输协议（TCP、SSL、HTTP 等）已及服务器连接参数（主机、端口等）。

[提交 bug 报告](#)

20.6.6. 配置 JMS 桥

HornetQ 包含一个具有完整功能的 JMS 消息桥。这个桥的功能是从源队列或主题消费消息，并发送消息到通常位于不同服务器上的目标队列或主题。

源和目标服务器不需要位于相同的群集，对于跨 WAN 的、连接不稳定的实例，这使得桥适合于从一个群集可靠地发送消息至另外一个群集。

桥也可以部署为 HornetQ 独立服务器上的独立应用程序，或者位于 JBoss AS 实例内部。源和目标服务器既可以位于相同的、也可以位于不同的虚拟机里。

例 20.4. JMS 桥的配置示例：

这个例子中的值适用于本节的剩余内容。

```
<subsystem>
  <subsystem xmlns="urn:jboss:domain:messaging:1.3">
    <hornetq-server>
      ...
    </hornetq-server>

    <jms-bridge name="myBridge">
      <source>
```

```

        <connection-factory name="ConnectionFactory"/>
        <destination name="jms/queue/InQueue"/>
    </source>
    <target>
        <connection-factory
name="jms/RemoteConnectionFactory"/>
        <destination name="jms/queue/OutQueue"/>
        <context>
            <property key="java.naming.factory.initial"
value="org.jboss.naming.remote.client.InitialContextFactory"/>
            <property key="java.naming.provider.url"
value="remote://192.168.40.1:4447"/>
        </context>
    </target>
    <quality-of-service>AT_MOST_ONCE</quality-of-service>
    <failure-retry-interval>1000</failure-retry-interval>
    <max-retries>-1</max-retries>
    <max-batch-size>10</max-batch-size>
    <max-batch-time>100</max-batch-time>
    <add-messageID-in-header>true</add-messageID-in-header>
</jms-bridge>

...
</subsystem>

```

表 20.9. HornetQ Core JMS 属性

属性	描述
name	服务器上所有的桥都必须有一个唯一名称。
source connection-factory	它注入 SourceCFF bean（也定义在 Bean 文件里）。这个 bean 创建源 ConnectionFactory。
source destination name	它注入 SourceDestinationFactory bean（也定义在 Bean 文件里）。这个 bean 创建源 Destination。
target connection-factory	它注入 TargetCFF bean（也定义在 Bean 文件里）。这个 bean 创建目标 ConnectionFactory。
target destination name	它注入 TargetDestinationFactory bean（也定义在 Bean 文件里）。这个 bean 创建目标 Destination。
quality-of-service	这个参数代表服务模式要求的品质。可能的值包括 AT_MOST_ONCE、DUPLICATES_OK、ONCE_AND_ONLY_ONCE。
failure-retry-interval	它表示当桥检测到与源或目标服务器的连接已失败时重新创建连接前等待的时间（毫秒）。
max-retries	它表示当桥检测到与源或目标服务器的连接已失败时重新创建连接的次数。桥在尝试这个次数后会放弃。-1 表示“无限次尝试”。

属性	描述
max-batch-size	它表示在以批次将消息发送到目标目的地之前从源目的地消费的消息的最大数目。它的值必须大于 1。
max-batch-time	它表示在发送批次消息到目标目的地之前等待的时间（毫秒），即使已消费的消息数量还没有到达 MaxBatchSize 。它的值为 -1 表示“一直等待”，大于 1 则指定实际的时间。
add-messageID-in-header	<p>如果为 true，原始消息的 ID 将附加在发送到目的地的消息头部 HORNETQ_BRIDGE_MSG_ID_LIST。如果消息被多次桥接，每个消息 ID 都会被附加。这启用了要使用的分布式请求-响应模式。</p> <p>当您接收消息时，您可以用第一个消息 ID 的 correlation id 发送响应，所以当原始发送者接收到这个消息时，它就容易进行关联。</p>

关于更完整的说明，请参考 [第 20.9.2 节“创建 JMS 桥”](#)。

[提交 bug 报告](#)

20.6.7. 配置延迟的重递送

介绍

延迟的重递送是在 **<redelivery-delay>** 元素里定义的，它是 JMS 子系统配置的 **<address-setting>** 元素的子元素。

```
<!-- delay redelivery of messages for 5s -->
<address-setting match="jms.queue.exampleQueue">
  <redelivery-delay>5000</redelivery-delay>
</address-setting>
```

如果指定了重递送时间，JMS 系统在重新递送这些消息前将等待。如果 **<redelivery-delay>** 被设置为 **0**，将不会发送重递送。地址通配符可以用在 **<address-match>** 元素的 **match** 属性上来配置匹配通配符的地址的重递送延迟。

[提交 bug 报告](#)

20.6.8. 配置 Dead Letter 地址

介绍

Dead Letter 地址是在 JMS 子系统配置的 **<address-setting>** 元素里定义的。

```
<!-- undelivered messages in exampleQueue will be sent to the dead letter
address
deadLetterQueue after 3 unsuccessful delivery attempts
-->
<address-setting match="jms.queue.exampleQueue">
  <dead-letter-address>jms.queue.deadLetterQueue</dead-letter-address>
  <max-delivery-attempts>3</max-delivery-attempts>
</address-setting>
```


■

如果没有指定 `<dead-letter-address>`，消息在试图递送 `<max-delivery-attempts>` 次后将被删除。在默认情况下，消息会尝试递送 10 次。设置 `<max-delivery-attempts>` 为 `-1` 会无限期地进行重递送。例如，对于一系列匹配的地址可以设置一个全局 Dead letter，而且对于专有的地址设置可以将 `<max-delivery-attempts>` 设置为 `-1` 来允许无限期的重递送到这个地址。地址通配符也可以用来配置一系列地址的 Dead Letter 设置。

[提交 bug 报告](#)

20.6.9. 配置消息过期地址

介绍

消息过期地址是在 JMS 的 `address-setting` 配置里定义的。例如：

```
<!-- expired messages in exampleQueue will be sent to the expiry address
expiryQueue -->
<address-setting match="jms.queue.exampleQueue">
  <expiry-address>jms.queue.expiryQueue</expiry-address>
</address-setting>
```

如果消息已过期且没有指定过期地址，消息将从队列里删除并丢弃。*地址通配符 (Address wildcards)* 也可以用来配置一系列地址的过期地址范围。关于 JMX 通配符语法和示例，请参考 [第 20.6.2 节“配置 JMS 地址设置”](#)。

[提交 bug 报告](#)

20.6.10. 对 HornetQ 配置属性的引用

HornetQ 的 JBoss EAP 6 实现开放了下列可配置的属性。你可以使用管理 CLI 通过 `read-resource` 操作开放可配置或可查看的属性。

例 20.5. 实例

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-
server=default:read-resource
```

表 20.10. HornetQ 属性

属性	默认值	类型	描述
allow-failback	true	BOOLEAN	如果原来的在线服务器重新上线，是否自动关闭这个服务器。
async-connection-execution-enabled	true	BOOLEAN	服务器上的转入数据包是否必须移交线程池里的线程进行处理。

属性	默认值	类型	描述
address-setting			地址设置定义了按照地址通配符而不是专有队列定义的一些属性。
acceptor			接收器（Acceptor）定义 HornetQ 服务器接受哪些连接的方式。
backup-group-name		STRING	一系列必须彼此复制的 live/backup 服务器的名称
backup	false	BOOLEAN	服务器是否是备份（backup）服务器
check-for-live-server	false	BOOLEAN	被复制的在线服务器是否必须检查当前的群集来查看是否已有具有相同节点 ID 的在线服务器。
clustered	false	BOOLEAN	[已舍弃] 服务器是否加入群集
cluster-password	CHANGE ME!!	STRING	群集连接用来在群集节点间通讯的密码
cluster-user	HORNETQ.CLUSTER.ADMIN.USER	STRING	群集连接用来在群集节点间通讯的用户
cluster-connection			群集连接将服务器分组为群集，这样消息就可以在群集节点间实现负载均衡。
create-bindings-dir	true	BOOLEAN	服务器在启动时是否必须创建 bindings 目录
create-journal-dir	true	BOOLEAN	服务器在启动时是否必须创建 journal 目录
connection-ttl-override	-1L	LONG	如果设置，它将覆盖连接保持在线而无需要接收 ping 的时间（毫秒）。
connection-factory			定义一个连接工厂
connector			客户用来定义如何连接服务器的连接器

属性	默认值	类型	描述
connector-service			
divert			这是一个消息资源，它允许您将路由至某个地址的消息透明地转移到其他地址，而无需修改任何客户应用程序逻辑。
discovery-group			通过侦听从声明连接器的其他服务器接收广播信息的多点传送组
failback-delay	5000	LONG	在线服务器重启时发生故障恢复前等待的时间
failover-on-shutdown	false	BOOLEAN	在普通的服务器关闭时，这个备份服务器（如果是）是否必须在线。
grouping-handler			决定群集里的哪个节点必须处理带有分配的组 ID 的消息
id-cache-size	20000	INT	用于预先创建消息 ID 的缓存的大小
in-vm-acceptor			定义 HornetQ 服务器接受哪些 in-VM 连接的方式
in-vm-connector			in-VM 客户用来定义如何连接服务器
jmx-domain	org.hornetq	STRING	用来在 MBeanServer 里注册内部 HornetQ MBean 的 JMX 域
jmx-management-enabled	false	BOOLEAN	HornetQ 是否必须通过 JMX 开放其内部的管理 API。这不是我们推荐的做法，因为访问这些 MBean 可能导致不一致的配置。
journal-buffer-size	501760 (490KiB)	LONG	日志上的内部缓冲的大小

属性	默认值	类型	描述
journal-buffer-timeout	对于 ASYNCIO 日志, 500000 (0.5 毫秒), 对于 NIO 日志, 3333333 (3.33 毫秒)	LONG	用于冲刷日志内部缓冲的间隔 (纳秒)
journal-compact-min-files	10	INT	在可以开始压缩前最少的日志数据文件数目
journal-compact-percentage	30	INT	我们考虑压缩日志前活动数据所占的百分比
journal-file-size	10485760	LONG	每个日志文件的大小 (字节)
journal-max-io	1	INT	任一时刻 AIO 队列里写请求的最大数量。当使用 ASYNCIO 日志时默认值修改为 500。
journal-min-files	2	INT	要预先创建的日志文件的数量
journal-sync-non-transactional	true	BOOLEAN	在向客户返回响应之前是否等待非事务数据同步至日志
journal-sync-transactional	true	BOOLEAN	在向客户返回响应之前是否等待事务数据同步至日志
journal-type	ASYNCIO	字符串	要使用的日志的类型。这个属性的值可以是 "ASYNCIO" 或 "NIO"。
jms-topic			定义 JMS 主题
live-connector-ref	reference	STRING	[已舍弃] 用来连接在线服务的连接器的名称。如果这个服务器不是使用共享 HA 的备份服务器, 它的值为 "undefined"
log-journal-write-rate	false	BOOLEAN	是否定期记录日志的写速率和冲刷速率

属性	默认值	类型	描述
mask-password	true	BOOLEAN	
management-address	jms.queue.hornetq.management	STRING	管理消息送往的地址
management-notification-address	hornetq.notifications	STRING	消费者绑定以接收管理通知的地址的名称
max-saved-replicated-journal-size	2	INT	在故障恢复发生后保持的日志备份的最大数量
memory-measure-interval	-1	LONG	JVM 内存取样的频率（毫秒，-1 表示禁用内存取样）
memory-warning-threshold	25	INT	超过就会记录警告日志的可用内存的百分比
message-counter-enabled	false	BOOLEAN	是否启用消息计数
message-counter-max-day-history	10	INT	保持消息计数历史的天数
message-counter-sample-period	10000	LONG	用于消息计数的样本周期（毫秒）
message-expiry-scan-period	30000	LONG	扫描过期消息的频率（毫秒）
message-expiry-thread-priority	3	INT	线程过期消息的优先级

属性	默认值	类型	描述
page-max-concurrent-io	5	INT	分页允许的并发读取的最大数量
perf-blast-pages	-1	INT	
persist-delivery-count-before-delivery	false	BOOLEAN	是否在递送前持久化递送计数。 False 表示这只有在取消消息后才会发生。
persist-id-cache	true	BOOLEAN	是否将 ID 持久化到日志
persistence-enabled	true	BOOLEAN	服务器是否将使用基于文件的日志进行持久化
pooled-connection-factory			定义受管连接工厂
remoting-interceptors	undefined	LIST	[已舍弃] 这个服务器使用的拦截器类列表
remoting-incoming-interceptors	undefined	LIST	服务器使用的转入拦截器类的列表
remoting-outgoing-interceptors	undefined	LIST	服务器使用的转出拦截器类的列表
run-sync-speed-test	false	BOOLEAN	是否在启动时执行关于磁盘同步速度的诊断测试。当确认性能问题时这很有用。
replication-clustername		STRING	如果配置了多个群集连接，将复制的群集连接的名称。
runtime-queue			Runtime 队列

属性	默认值	类型	描述
remote-connector			远程客户用它来定义如何连接服务器
remote-acceptor			定义 HornetQ 服务器接受哪些远程连接的方式
scheduled-thread-pool-max-size	5	INT	主调度线程池拥有的线程数量
security-domain	other	STRING	用来检验用户和角色信息的安全域
security-enabled	true	BOOLEAN	是否启用安全性
security-setting			这是一个安全设置，它允许根据基于地址的队列定义权限集。
security-invalidati on-interval	10000	LONG	子安全缓存失效前等待的时间（毫秒）
server-dump-interval	-1	LONG	将运行时信息转储至服务器日志的频率。小于 1 的值表示禁用这个功能。
shared store	true	BOOLEAN	服务器是否将共享存储用于失效切换
thread-pool-max-size	30	INT	主线程池拥有的线程数量。-1 表示无限制。
transaction-timeout	300000	LONG	在创建后多久事务可以从资源管理者删除
transaction-timeout-scan-period	1000	LONG	扫描超时事务的频率（毫秒）
wild-card-routing-enabled	true	BOOLEAN	服务器是否支持通配符路由



警告

journal-file-size 的值必须比发往服务器的消息大小要大，否则服务器无法存储这个消息。

[提交 bug 报告](#)

20.6.11. 设置消息过期

介绍

如果发送的消息没有在指定时间（毫秒）内递送给消费者，它们可被设置为过期。使用 **Java** 消息服务（JMS）或 **HornetQ Core API**，您可以直接在消息上设置过期时间。例如：

```
// message will expire in 5000ms from now
message.setExpiration(System.currentTimeMillis() + 5000);
```

JMS MessageProducer 包含了一个 **TimeToLive** 参数，它控制发送的消息的过期：

```
// messages sent by this producer will be retained for 5s (5000ms) before
expiration
producer.setTimeToLive(5000);
```

从过期地址消费的过期消息具有下列属性：

- **_HQ_ORIG_ADDRESS**

包含过期消息的原始地址的字符串型属性

- **_HQ_ACTUAL_EXPIRY**

包含过期消息的实际过期时间的长整型属性。

配置消息过期地址

消息过期地址是在 **JMS** 的 **address-setting** 配置里定义的：

```
<!-- expired messages in exampleQueue will be sent to the expiry address
expiryQueue -->
<address-setting match="jms.queue.exampleQueue">
  <expiry-address>jms.queue.expiryQueue</expiry-address>
</address-setting>
```

如果消息已过期且没有指定过期地址，消息将从队列里删除并丢弃。

配置 Expiry Reaper 线程

Reaper 线程会定期地检查队列以检验消息是否过期。

- **message-expiry-scan-period**

扫描队列以检测过期消息的频率（毫秒为单位，默认是 30000 毫秒。-1 表示禁用 Reaper 线程）。

- `message-expiry-thread-priority`

Reaper 线程的优先级。它必须是 0 - 9 之间的值，9 表示最高优先级。默认值为 3。

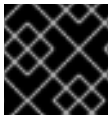
[提交 bug 报告](#)

20.7. 消息分组

20.7.1. 关于消息分组

消息组是一组共享某些特征的消息：

- 消息组里所有消息都按照通用的组 ID 进行分组。这意味着它们可以用通用的组属性来识别。
- 不管队列里有多少客户，消息组里的所有消息都被相同的消费者顺次进行处理和消费。这意味着如果具有唯一组 ID 的消息组被某个消费者打开，它总是由这个消费者来处理。如果消费者关闭了这个消息组，整个消息组都会被指引到队列里的其他消费者。



重要

当需要由单个消费者顺次处理具有某个属性值（如组 ID）的消息时，消息组就尤其有用。

[提交 bug 报告](#)

20.7.2. 在客户端使用 HornetQ Core API

属性 `_HQ_GROUP_ID` 用于在客户端识别 HornetQ Core API 里的消息组。要选取随机的唯一消息组 ID，您也可以在 `SessionFactory` 上设置 `autogroup` 属性为 "true"。

[提交 bug 报告](#)

20.7.3. 为 Java Messaging Service (JMS) 客户配置服务器

属性 `JMSXGroupID` 用于为 JMS 客户识别消息组。如果您想发送包含不同消息的消息组到某个消费者，您可以对不同的消息设置相同的 `JMSXGroupID`：

```
Message message = ...
message.setStringProperty("JMSXGroupID", "Group-0");
producer.send(message);

message = ...
message.setStringProperty("JMSXGroupID", "Group-0");
producer.send(message);
```

第二个途径是设置 `HornetQConnectonFactory` 上的 `autogroup` 属性为 "true"。

`HornetQConnectionFactory` 将选取一个随机的唯一消息组 ID。您可以这样在服务器配置文件（`standalone.xml` 和 `domain.xml`）里设置 `autogroup` 属性：

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty-connector"/>
```

```

</connectors>
<entries>
  <entry name="ConnectionFactory"/>
</entries>
<autogroup>true</autogroup>
</connection-factory>

```

除了上面两个途径以外，另外一个方法是通过连接工厂设置专有的消息组 ID。这将依次将通过这个连接工厂发送的所有消息的属性 **JMSXGroupID** 设置为指定的值。要在连接工厂上设置专有的消息组 ID，请编辑服务器配置文件（**standalone.xml** 和 **domain.xml**）里的 **group-id** 属性：

```

<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty-connector"/>
  </connectors>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
  <group-id>Group-0</group-id>
</connection-factory>

```

[提交 bug 报告](#)

20.7.4. 群集分组

群集分组遵循相对于普通消息分组的不同途径。在群集里，具有专有组 ID 的消息组可以达到任何节点。对于节点来说，确定哪个组 ID 绑定到哪个节点上的哪个消费者是很重要的。每个节点都负责将消息正确路由至有客户处理这些组 ID 的节点，而不管消息组默认到达什么地方。

这种情况可以用分组处理程序来解决。每个节点有一个分组处理程序，且这个分组处理程序（以及其他处理程序）负责将消息组路由至正确的节点。目前有两种类型的分组处理程序：**local** 和 **remote**。

Local 处理程序负责决定消息组应该使用的路由。**Remote** 处理程序与 **Local** 处理程序通讯并响应地进行工作。每个群集应该选取一个专有节点运行 **Local** 处理程序，而其他节点则运行 **Remote** 处理程序。

您可以在服务器配置文件（**standalone.xml** 和 **domain.xml**）里配置 "local" 和 "remote" 组处理程序：

```

<grouping-handler name="my-grouping-handler">
  <type>LOCAL</type>
  <address>jms</address>
  <timeout>5000</timeout>
</grouping-handler>

<grouping-handler name="my-grouping-handler">
  <type>REMOTE</type>
  <address>jms</address>
  <timeout>5000</timeout>
</grouping-handler>

```

"timeout" 属性确保路由决定在指定时间内快速完成。如果在这段时间内没有作出决定，异常将被抛出。

最初接收消息组的节点将根据常规的群集路由条件（**round-robin** 队列可用性）作出路由决定。节点不同的组处理程序将建议这个决定，如果它们接受建议，然后再将这些消息转发到提议的队列。

如果组处理程序拒绝这个建议，节点会建议其他路由。其他节点会遵循套件并将消息组转发到所选的队列。在消息到达队列后，它将锁定队列上的某个客户。

[提交 bug 报告](#)

20.7.5. 群集分组的最佳实践

下面是群集分组的一些最佳实践：

- 如果您经常创建和关闭消费者以确保消费者均匀地分布在不同的节点上。一旦锁定了某个队列，消息将自动地转移到该队列而不管是否删除它的客户。
- 如果您希望删除一个绑定了消息组的队列，请确保这个队列是由发送消息的会话删除的。这样可以确保在删除队列后，其它节点不会试图将消息路由至这个队列。
- 这是因为失效切换机制总是复制具有本地分组处理程序的节点。

[提交 bug 报告](#)

20.8. 重复消息的检测

20.8.1. 关于重复消息的检测

重复消息检测可以过滤重复消息而无需在应用程序里编写重复消息检测代码。您可以在 **HornetQ** 里配置重复消息检测。

当发送者（客户/服务器）发送消息到另外一个服务器时，有时候目标服务器（接收者）或连接在发送消息后但在发送者接收到响应以表示成功前发生故障。在这种情况下，发送者（客户）很难确定消息是否已经成功发送给预期的接收者。

发送的消息可能成功也可能不成功，这取决于目标接收者或连接何时出现故障（在发送消息之前或之后）。如果发送者（客户/服务器）决定重发最后一条消息，这可能会导致发送重复的消息。

HornetQ 为发送的消息提供了重复消息检测。

[提交 bug 报告](#)

20.8.2. 对消息发送使用重复消息检测

要启用重复消息检测，您需要将消息上的一个特殊属性设置为唯一的值。您可以随意创建这个值，但它必须是唯一的。

当目标服务器接收到这个消息时，它会检查这个特殊属性是否已被设置。如果这个属性已设置，目标服务器将根据头部信息检查其内存缓存里的消息。如果服务器找到具有任何相同头部信息的消息，它将忽略这些客户发送的消息。

如果您发送事务里的消息，您不需要为每条消息设置属性；您只需要在事务里设置一次就可以了。如果服务器检测到事务里有重复的消息，它会忽略整个事务。

您设置的属性的名称是由 `org.hornetq.api.core.HDR_DUPLICATE_DETECTION_ID` 的值指定的，它是 `_HQ_DUPL_ID`。对于 Core API，这个属性的值的类型可以是 `byte[]` 或 `SimpleString`。对于 JMS 客户，它的类型必须是 `String` 且指定唯一的值。UUID 是生成唯一 ID 的简单途径。

下面的例子展示了如何为 Core API 设置这个属性：

```
...

ClientMessage message = session.createMessage(true);

SimpleString myUniqueID = "This is my unique id";    // Can use a UUID for
this

message.setStringProperty(HDR_DUPLICATE_DETECTION_ID, myUniqueID);

...
```

下面的例子展示了如何为 JMS 客户设置这个属性：

```
...

Message jmsMessage = session.createMessage();

String myUniqueID = "This is my unique id";    // Could use a UUID for this

message.setStringProperty(HDR_DUPLICATE_DETECTION_ID.toString(),
myUniqueID);

...
```

[提交 bug 报告](#)

20.8.3. 配置重复 ID 缓存

服务器维护着发送到每个地址的

org.hornetq.core.message.impl.HDR_DUPLICATE_DETECTION_ID 属性值的缓存。每个地址也都维护着自己的地址缓存。

缓存的大小是固定的。缓存的最大尺寸可通过服务器配置文件 (**standalone.xml** 和 **domain.xml**) 里的 ***id-cache-size*** 参数进行配置。这个参数的默认值是 2000 个元素。如果缓存最大尺寸为 N，那么存储的第 N+1 个 ID 将覆盖缓存里的第 0 个元素。

您也可以配置缓存是否持久化到磁盘。这可以用服务器配置文件 (**standalone.xml** 和 **domain.xml**) 里的 ***persist-id-cache*** 参数来配置。如果这个值为 **true**，那么每个接收到的 ID 都会持久化到永久性存储里。它的默认值是 **true**。



注意

将复制 ID 缓存的大小设置为较大值可以确保重发的消息不会覆盖缓存里之前发送的消息。

[提交 bug 报告](#)

20.8.4. 对桥和群集连接使用重复消息检测

您可以配置核心桥在转发消息至目标前自动添加唯一的重复 ID 值（如果还没有）。要配置核心桥的重复消息检测，请在服务器配置文件 (**standalone.xml** 和 **domain.xml**) 里将属性 ***use-duplicate-***

detection 设置为 "true"。这个参数的默认值是 "true"。

群集连接在内部使用核心桥在群集节点间移动消息。要配置群集连接的重复消息检测，请在服务器配置文件 (**standalone.xml** 和 **domain.xml**) 里将属性 **use-duplicate-detection** 设置为 "true"。这个参数的默认值是 "true"。

[提交 bug 报告](#)

20.9. JMS 桥

20.9.1. 关于消息桥 (Bridge)

桥的功能是从源队列消费消息，并将其转发到通常位于不同 HornetQ 服务器上的目标地址。桥处理不可靠的连接，当连接再次可用时自动重连。HornetQ 桥可以用过滤器表达式进行配置，只转发某些消息。



重要

JMS 桥不能部署至 EAP 6 服务器里，这包括作为指定备份服务器配置的 HornetQ。其原因是指定备份服务器上的事务管理者不能恢复之前在 HornetQ 在线服务器上开始的事务。

[提交 bug 报告](#)

20.9.2. 创建 JMS 桥

介绍

JMS 桥消费源 JMS 队列或主题里的消息并发送到目标 JMS 队列或主题（通常位于不同的服务器上）。它可以用于桥接任何 JMS 服务器间的消息，只要这些消息是兼容 JMS 1.1 的。源和目的 JMS 资源通过 JNDI 查找，用于 JNDI 查找的客户类必须捆绑在模块里。然后在 JMS 桥配置里声明模块名。

过程 20.2. 创建 JMS 桥

这个过程演示了如何配置 JMS 桥从 JBoss EAP 5.x 移植消息到 JBoss EAP 6 服务器。

1. 在源 JMS 消息服务器上配置桥

请使用源服务器提供的说明配置 JMS 桥。关于如何在 JBoss EAP 5.x 上配置 JMS 桥的例子，请参考《JBoss EAP 6 移植指南》里的 *Create a JMS Bridge* 章节。

2. 配置目的 JBoss EAP 6 服务器上的 JMS 桥

在 JBoss EAP 6.1 以后的版本里，JMS 桥可以用于从任何兼容 JMS 1.1 的服务器上桥接消息。因为源和目标 JMS 资源是用 JNDI 进行查找的，源消息供应商或消息中介的 JNDI 查找类必须捆绑在 JBoss 模块里。下面的过程使用了虚拟的 'MyCustomMQ' 消息中介作为例子。

a. 为消息供应商创建 JBoss 模块。

- i. 在 **EAP_HOME/modules/system/layers/base/** 下为新的模块创建一个目录结构。**main/** 子目录将包含客户 JAR 和 **module.xml** 文件。下面是为消息供应商 MyCustomMQ 创建的一个目录结构示例：**EAP_HOME/modules/system/layers/base/org/mycustommq/main/**。

- ii. 在 **main/** 子目录里，创建一个包含消息供应商的模块定义的 **module.xml** 文件。下面是为消息供应商 MyCustomMQ 创建的 **module.xml** 示例。

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<module xmlns="urn:jboss:module:1.1" name="org.mycustommq">
  <properties>
    <property name="jboss.api" value="private"/>
  </properties>

  <resources>
    <!-- Insert resources required to connect to the
    source or target -->
    <resource-root path="mycustommq-1.2.3.jar" />
    <resource-root path="mylogapi-0.0.1.jar" />
  </resources>

  <dependencies>
    <!-- Add the dependencies required by JMS Bridge code
    -->
    <module name="javax.api" />
    <module name="javax.jms.api" />
    <module name="javax.transaction.api"/>
    <!-- Add a dependency on the org.hornetq module since
    we send -->
    <!-- messages to the HornetQ server embedded in the
    local EAP instance -->
    <module name="org.hornetq" />
  </dependencies>
</module>

```

- iii. 从源资源复制消息供应商用于 JNDI 查找的 JAR 到模块的 **main/** 子目录。
MyCustomMQ 模块的目录结构应该类似于：

```

modules/
  -- system
    -- layers
      -- base
        -- org
          -- mycustommq
            -- main
              -- mycustommq-1.2.3.jar
              -- mylogapi-0.0.1.jar
              -- module.xml

```

- b. 配置部署到 JBoss EAP 6 服务器的 **messaging** 子系统的 JMS 桥。

- i. 在开始之前，请先停止服务器并备份当前的服务器配置文件。如果是作为独立服务器运行，它是 **EAP_HOME/standalone/configuration/standalone-full-ha.xml**。如果是运行的受管域，请备份 **EAP_HOME/domain/configuration/domain.xml** 以及 **EAP_HOME/domain/configuration/host.xml**。
- ii. 在服务器配置文件里的 **messaging** 子系统里添加 **jms-bridge** 元素。**source** 和 **target** 元素提供的用于 JNDI 查找的 JMS 资源的名称。如果指定了 **user** 和 **password** 凭证，当创建 JMS 连接时它们会作为参数传入。

下面是一个为消息供应商 MyCustomMQ 配置的 **jms-bridge** 元素例子：

```

<subsystem xmlns="urn:jboss:domain:messaging:1.3">

```



```

...
<jms-bridge name="myBridge" module="org.mycustommq">
  <source>
    <connection-factory name="ConnectionFactory"/>
    <destination name="sourceQ"/>
    <user>user1</user>
    <password>pwd1</password>
    <context>
      <property key="java.naming.factory.initial"
value="org.mycustommq.jndi.MyCustomMQInitialContextFactory"/>
      <property key="java.naming.provider.url"
value="tcp://127.0.0.1:9292"/>
    </context>
  </source>
  <target>
    <connection-factory name="java:/ConnectionFactory"/>
    <destination name="/jms/targetQ"/>
  </target>
  <quality-of-service>DUPLICATES_OK</quality-of-service>
  <failure-retry-interval>500</failure-retry-interval>
  <max-retries>1</max-retries>
  <max-batch-size>500</max-batch-size>
  <max-batch-time>500</max-batch-time>
  <add-messageID-in-header>true</add-messageID-in-header>
</jms-bridge>
</subsystem>

```

在上面的例子里，JNDI 属性是在 **source** 的 **context** 元素里定义的。如上面的 **target** 例子，如果忽略了 **context** 元素，JMS 资源将在本地实例里进行查找。

[提交 bug 报告](#)

20.10. 持久化

20.10.1. 关于 HornetQ 里的持久化

HornetQ 处理自己的持久化。它附带为专有消息用例进行了优化的高性能日志。

HornetQ 日志只使用可配置的文件大小进行附加，它通过单个写入操作来提高性能。它包含一系列文件，开始时都以固定大小预先创建并填充空白内容。当执行服务器操作（**add message**、**delete message**、**update message** 等）时，操作记录会附加到日志后面，直至日志文件已满，此时将使用下一个日志文件。

复杂的垃圾收集算法确定日志文件在所有数据被删除后是否被回收并重用。压缩算法将从日志文件删除无效空间并压缩数据。

日志也完全支持本地和 XA 事务。

日志的主要部分是用 Java 编写的，但和文件系统的交互已经抽象化为不同的可插入实现。HornetQ 附带的两个实现是：

- *Java New I/O (NIO)*

使用标准的 Java NIO 与文件系统交互。这提供了极佳的性能且可以运行在任何装有 Java 6 或更高版本的平台上。

- *Linux Asynchronous IO (AIO)*

使用 Native Code Wrapper 和 Linux Asynchronous IO Library (AIO) 进行交互。通过 AIO, HornetQ 在数据已被持久化时接收消息。这避免了对显性同步的需要。AIO 通常会提供比 Java NIO 更好的性能,但它要求 Linux kernel 2.6 或更高的版本以及 libaio 软件包。

AIO 也要求 ext2、ext3、ext4、jfs 或 xfs 文件系统。

标准的 HornetQ 核心服务器使用下列日志实例：

- *bindings journal*

存储和绑定相关的数据,包括部署在服务器上的队列及其属性。它也存储 ID 序列计数器等数据。绑定日志总是 NIO 日志,因为和消息日志相比它通常具有较低的吞吐量。

这个日志里的文件的前缀是 **hornetq-bindings**。每个文件都有一个绑定扩展名。文件大小是 1048576 字节,它位于 **bindings** 文件夹里。

- *JMS journal*

存储所有和 JMS 相关的数据,如任何 JMS 队列、主题或连接工厂以及用于这些资源的 JNDI 绑定。用管理 API 创建的任何 JMS 资源都保存在这个日志里,而用配置文件配置的资源则不会。这个日志只有在使用 JMS 时才会被创建。

- *message journal*

存储所有和消息相关的数据,包括消息本身及 **duplicate-id** 缓存。在默认情况下, HornetQ 将 AIO 用于这个日志。如果 AIO 不可用,它将自动退回为 NIO。

大型消息是在消息日志之外持久化的。在内存低的环境里,可以配置 HornetQ 将消息页面化到磁盘上。如果不需要持久化,可以配置 HornetQ 不持久化任何数据。

[提交 bug 报告](#)

20.11. HORNETQ 群集

HornetQ 群集用于创建 HornetQ 服务器组来分享消息处理的负载。群集里每个活动节点都充当独立的 HornetQ 服务器并管理自己的消息和连接。

要组成群集,每个节点(独立的 HornetQ 服务器)都用服务器配置文件(**standalone.xml** 和 **domain.xml**)的配置参数声明到其它节点的群集连接。

在群集里,核心桥(Core Bridge)用来将消息从一个群集桥接/路由到另外一个群集。核心桥从源队列消费消息并转发这些消息至可能位于相同群集或不同群集的目标 HornetQ 服务器(节点)。

当节点和其它节点组成一个群集连接时,它会在内部创建一个核心桥。每个节点都创建一个显性的核心桥,您并不需要声明它。这些群集连接允许消息在平衡消息处理负载的不同群集的节点间传输。

您可以在服务器配置文件(**standalone.xml** 和 **domain.xml**)里配置群集节点。



重要

您可以通过服务器配置文件 (**standalone.xml** 和 **domain.xml**) 来配置节点并将这个配置复制到其它节点以生成对称型群集。然而，当复制服务器配置文件是您必须小心。您不能复制 **HornetQ** 数据 (如绑定、日志和大型消息目录)。当节点第一次启动时，它会将正确格式化群集所需的唯一标识符存储在日志目录。

[提交 bug 报告](#)

20.11.1. 关于服务器发现

服务器使用一个名为“服务器发现 (server discovery)”的机制来实现：

- 转发它们的连接细节到消息客户：消息客户愿意连接群集里的服务器而无需在给定时间点在线和运行的服务器的细节。
- 连接至其它服务器：群集里的服务器想建立与其它服务器的群集连接而无需群集里所由其它服务器的专有细节。

服务器的信息通过普通的 **HornetQ** 连接发送给消息客户并通过群集连接发送给其它服务器。

这需要建立初始的连接，它可以通过动态的服务器发现技术如 **UDP**、**JGroups** 或者根据连接器列表来建立。

[提交 bug 报告](#)

20.11.2. 广播组

客户使用连接器来定义连接至服务器的方式和途径。服务器使用广播组 (**broadcast group**) 来在网络里广播连接器。广播组使用一系列连接器对并在网络上进行广播。每个连接器对都包含用于在线和备份服务器的连接设置。

您可以在服务器配置文件 (**standalone.xml** 和 **domain.xml**) 里的 **broadcast-groups** 定义广播组。单个 **HornetQ** 服务器可以有多个广播组。您也可以定义用户数据报文协议 (**User Datagram Protocol**, **UDP**) 或 **JGroup** 广播组。

[提交 bug 报告](#)

20.11.2.1. UDP 广播组

下面的例子定义了一个 **UDP** 广播组：

```
<broadcast-groups>
  <broadcast-group name="my-broadcast-group">
    <local-bind-address>172.16.9.3</local-bind-address>
    <local-bind-port>5432</local-bind-port>
    <group-address>231.7.7.7</group-address>
    <group-port>9876</group-port>
    <broadcast-period>2000</broadcast-period>
    <connector-ref>netty</connector-ref>
  </broadcast-group>
</broadcast-groups>
```



注意

在上面的配置示例里，属性"local-bind-address"、"local-bind-port"、"group-address"和 "group-port" 都已舍弃不用。您可以选择使用 "socket-binding" 属性。

下面的例子定义了一个 UDP 广播组，它用属性 "socket-binding" 替换了所有已舍弃不用的属性。

```
<broadcast-groups>
  <broadcast-group name="my-broadcast-group">
    <socket-binding>messaging-group</socket-binding>
    <broadcast-period>2000</broadcast-period>
    <connector-ref>netty</connector-ref>
  </broadcast-group>
</broadcast-groups>
```

下表描述了上例里使用的常用来定义 UDP 广播组的重要参数：

表 20.11. UDP 广播组参数

属性	描述
name attribute	指定服务器里每个广播组的名称。每个广播组都必须有一个唯一名称。
local-bind-address	[已舍弃] 这是 UDP 的专有属性，它用来指定数据报文套接字绑定的本地地址。您必须设置这个属性以定义用于广播的接口。如果没有指定这个属性，那么套接字会绑定到配符地址（内核随机生成的地址）。
local-bind-port	[已舍弃] 这是 UDP 的专有属性，它用来指定数据报文套接字绑定的本地端口。默认值 "-1" 表示使用匿名的端口。
group-address	[已舍弃] 这是广播消息时 UDP 专有的多点传送地址。这个 IP 地址的范围是 224.0.0.0 到 239.255.255.255（包含）。224.0.0 是保留地址且不能使用。
group-port	[已舍弃] 它指定了用于广播的 UDP 端口号
socket-binding	它指定了广播组的套接字绑定
broadcast-period	这个参数指定两次广播的时间间隔（毫秒）。它是可选参数。
connector-ref	它引用将被广播的连接器的。

[提交 bug 报告](#)

20.11.2.2. JGroups 广播组

您可以指定 JGroups 的两个属性来进行广播：*jgroups-stack* 和 *jgroups-channel*。下面的例子定义了一个 JGroups 广播组：

```
<broadcast-groups>
  <broadcast-group name="bg-group1">
    <jgroups-stack>udp</jgroups-stack>
    <jgroups-channel>udp</jgroups-channel>
    <broadcast-period>2000</broadcast-period>
    <connector-ref>netty</connector-ref>
  </broadcast-group>
</broadcast-groups>
```

JGroups 广播组定义使用了两个主要的属性：

- *jgroups-stack*：这个属性指定了 `org.jboss.as.clustering.jgroups` 子系统里定义的栈的名称。
- *jgroups-channel*：这个属性指定了 JGroups 频道广播时所连接的频道。

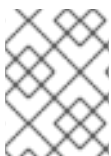
[提交 bug 报告](#)

20.11.3. 发现组

广播组用于在网络上广播连接器。而另一方面，发现组（Discovery Group）定义如何从广播端点（UDP 或 JGroups 广播组）获取连接器信息。发现组维护一个连接器对的列表 - 每个条目都对应不同服务器的广播。

当发现组接收到来自某个服务器的广播端点上的广播，它会相应地更新列表里的连接器对条目。如果长时间没有从某个服务器接收广播，它会将这个服务器的条目从列表里删除。

群集连接和 JMS 客户使用发现组主要是获取初始连接信息以下载所需的拓扑结构。



注意

您必须为每个发现组配置一个合适的匹配其对应的广播组（UDP 或 JGroups）的广播端点。

[提交 bug 报告](#)

20.11.3.1. 配置服务器上的 UDP 发现组

下面的例子定义了一个 UDP 发现组（Discovery Group）：

```
<discovery-groups>
  <discovery-group name="my-discovery-group">
    <local-bind-address>172.16.9.7</local-bind-address>
    <group-address>231.7.7.7</group-address>
    <group-port>9876</group-port>
    <refresh-timeout>10000</refresh-timeout>
  </discovery-group>
</discovery-groups>
```



注意

在上面的配置示例里，属性"local-bind-address"、"group-address" 和 "group-port" 都已舍弃不用。您可以选择使用 "socket-binding" 属性。

下面的例子定义了一个 UDP 发现组，它用属性 "socket-binding" 替换了所有已舍弃不用的属性。

```
<discovery-groups>
  <discovery-group name="my-discovery-group">
    <socket-binding>messaging-group</socket-binding>
    <refresh-timeout>10000</refresh-timeout>
  </discovery-group>
</discovery-groups>
```

下表描述了上例里使用的常用来定义 UDP 发现组的重要参数：

表 20.12. UDP 发现组参数

属性	描述
name	这个属性表示发现组的名称。对于每个服务器，每个发现名称都必须是唯一的。
local-bind-address	[已舍弃] 这是一个可选的 UDP 专有属性。在相同主机的多个接口时，它用于配置侦听专有接口的发现组。
group-address	[已舍弃] 这是一个强制的 UDP 专有属性。它用于配置侦听组的多点传送 ID 地址的发现组。这个属性的值必须匹配您要侦听的广播组的 group-address 属性。
group-port	[已舍弃] 这是一个强制的 UDP 专有属性。它用于配置多点传送组的 UDP 端口。这个属性的值必须匹配您要侦听的多点传送组的 group-port 属性。
socket-binding	它表示发现组的套接字绑定
refresh-timeout	这是一个可选的 UDP 专有属性。它用于配置发现组从某个服务器接收到最后一次广播后从列表里删除该服务器的连接器对条目前等待的时间（单位为毫秒）。 refresh-timeout 的值必须比广播组的 broadcast-period 的值高很多以防止在广播进程仍在时过早地删除服务器条目。这个属性的默认值是 10,000 毫秒。

[提交 bug 报告](#)

20.11.3.2. 配置服务器上的 JGroups 发现组

下面的例子定义了一个 JGroups 发现组（Discovery Group）：

```
<discovery-groups>
  <discovery-group name="dg-group1">
```

```

<jgroups-stack>udp</jgroups-stack>
<jgroups-channel>udp</jgroups-channel>
<refresh-timeout>10000</refresh-timeout>
</discovery-group>
</discovery-groups>

```

JGroups 发现组定义使用了两个主要的属性：

- **jgroups-stack**：这个属性指定了 `org.jboss.as.clustering.jgroups` 子系统里定义的栈的名称。
- **jgroups-channel**：这个属性指定了 JGroups 频道接收广播时所连接的频道。



注意

JGroup 属性和 UDP 专有属性不是互斥的。您可以在发现组或广播组的配置里使用 JGroup 或 UDP 属性集。

[提交 bug 报告](#)

20.11.3.3. 为 Java Messaging Service (JMS) 客户配置发现组

您可以为 JMS 和核心客户端程序配置发现组。您可以在服务器配置文件 (`standalone.xml` 和 `domain.xml`) 里指定用于 JMS 连接工厂的发现组。

```

<connection-factory name="ConnectionFactory">
  <discovery-group-ref discovery-group-name="my-discovery-group"/>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
</connection-factory>

```

discovery-group-ref 元素用于指定发现组的名称。当客户应用程序从 JNDI 下载这个连接工厂并创建 JMS 连接时，这些连接将通过侦听在发现组配置里指定的多点传送地址在发现组维护的所有服务器上进行负载均衡。

如果您在使用 JMS 而不时 JNDI 来查找连接工厂时，您可以在创建 JMS 连接工厂时直接指定发现组参数：

```

final String groupAddress = "231.7.7.7";
final int groupPort = 9876;
ConnectionFactory jmsConnectionFactory =
HornetQJMSClient.createConnectionFactory(new
DiscoveryGroupConfiguration(groupAddress, groupPort, new
UDPBroadcastGroupConfiguration(groupAddress, groupPort, null, -1)),
JMSFactoryType.CF);
Connection jmsConnection1 = jmsConnectionFactory.createConnection();
Connection jmsConnection2 = jmsConnectionFactory.createConnection();

```

refresh-timeout 属性的默认值可以通过 setter 方法 `setDiscoveryRefreshTimeout()` 在 `DiscoveryGroupConfiguration` 上进行设置。你可以用 `DiscoveryGroupConfiguration` 上的 `setDiscoveryInitialWaitTimeout()` 方法来指定连接工厂在创建第一个连接前等待的时间。

这样可以确保连接工厂有足够的时间来从群集里所有的节点接收广播。这个参数的默认值是 10,000 毫秒。

[提交 bug 报告](#)

20.11.3.4. 为 Core API 配置发现

如果您使用 Core API 来直接实例化 `ClientSessionFactory`，那么在创建会话工厂时您可以直接指定发现组参数：

```
final String groupAddress = "231.7.7.7";
final int groupPort = 9876;
ServerLocator factory = HornetQClient.createServerLocatorWithHA(new
DiscoveryGroupConfiguration(groupAddress, groupPort, new
UDPBroadcastGroupConfiguration(groupAddress, groupPort, null, -1)));
ClientSessionFactory factory = locator.createSessionFactory();
ClientSession session1 = factory.createSession();
ClientSession session2 = factory.createSession();
```

refresh-timeout 属性的默认值可以通过 setter 方法 `setDiscoveryRefreshTimeout()` 在 `DiscoveryGroupConfiguration` 上进行设置。在创建会话前，您可以使用会话工厂的 `DiscoveryGroupConfiguration` 上的 `setDiscoveryInitialWaitTimeout()` 方法等待指定的时间。

[提交 bug 报告](#)

20.11.4. 服务器端的负载平衡

重要的群集拓扑结构：

- 对称型群集：在对称型群集里，每个群集节点都直接连接到群集里其它节点。要创建对称型群集，群集里的每个节点都要定义一个群集连接并将属性 **max-hops** 设置为 1。



注意

在对称型群集里，每个节点都知道存在于其它节点上的所有队列及它们拥有哪些消费者。因此，群集可以确定在节点间如何平衡负载及重新分发消息。

[提交 bug 报告](#)

20.11.4.1. 配置群集连接

群集连接是在服务器配置文件 (`standalone.xml` 和 `domain.xml`) 的 **cluster-connection** 元素里定义的。每个 HornetQ 服务器可以定义零或多个群集连接。

```
<cluster-connections>
  <cluster-connection name="my-cluster">
    <address>jms</address>
    <connector-ref>netty-connector</connector-ref>
    <check-period>1000</check-period>
```

```

<connection-ttl>5000</connection-ttl>
<min-large-message-size>50000</min-large-message-size>
<call-timeout>5000</call-timeout>
<retry-interval>500</retry-interval>
<retry-interval-multiplier>1.0</retry-interval-multiplier>
<max-retry-interval>5000</max-retry-interval>
<reconnect-attempts>-1</reconnect-attempts>
<use-duplicate-detection>true</use-duplicate-detection>
<forward-when-no-consumers>false</forward-when-no-consumers>
<max-hops>1</max-hops>
<confirmation-window-size>32000</confirmation-window-size>
<call-failover-timeout>30000</call-failover-timeout>
<notification-interval>1000</notification-interval>
<notification-attempts>2</notification-attempts>
  <discovery-group-ref discovery-group-name="my-discovery-group"/>
</cluster-connection>
</cluster-connections>

```

下表定义了可配置属性：

表 20.13. 群集连接的可配置属性

属性	描述	默认值
<i>address</i>	每个群集连接都只适用于发送到以这个值开始的地址的消息。这个地址可以是任何值，您可以有许多具有不同地址值的群集连接同时在这些地址间或不同的群集服务器间平衡消息负载。这个值不使用通配符匹配。	
<i>connector-ref</i>	这是一个强制属性，它引用发送到群集里其他节点使其具有正确群集拓扑结构的连接器。	
<i>check-period</i>	它指定用来检验群集连接从其他服务器接收 ping 是否失败的时间周期（毫秒）。	30,000 毫秒
<i>connection-ttl</i>	它指定如果群集连接停止从群集里的特定节点接收消息，群集连接必须保持在线的时间。	60,000 毫秒
<i>min-large-message-size</i>	如果消息大小（字节）大于这个值，在通过网络发送到其他群集成员时它将被分成多个片段。	102,400 毫秒
<i>call-timeout</i>	它指定通过群集连接发送的数据包在抛出异常前等待回应的时间（毫秒）。	30,000 毫秒

属性	描述	默认值
<i>retry-interval</i>	如果在群集节点间创建群集连接，而目标节点还没有启动或正在重启，那么其他节点的群集连接将重新连接目标节点，直至它恢复。 <i>retry-interval</i> 定义了重试之间的时间间隔（毫秒）。	500 毫秒
<i>retry-interval-multiplier</i>	这个值是每次重试后的时间间隔 <i>retry-interval</i> 的增量。	1
<i>max-retry-interval</i>	它指定重试的最长延迟（毫秒）	2,000 毫秒
<i>reconnect-attempts</i>	它定义系统试图连接群集里节点的次数	-1（无限次重试）
<i>use-duplicate-detection</i>	群集连接使用桥来链接节点，您可以配置桥在转发的每条消息里添加一个 Duplicate ID。如果桥的目标节点崩溃后并恢复，消息可能从源节点再次发送。启用了重复检测后，目标节点将过滤和忽略任何重复的消息。	True
<i>forward-when-no-consumers</i>	这个参数指定消息是否将用 Round Robin 算法在群集的其他节点间分布而不管它们是否匹配其他节点上任何消费者。	False
<i>max-hops</i>	它指定消息如何在连接到这个服务器的其他 HornetQ 服务器间进行负载平衡。	-1
<i>confirmation-window-size</i>	用于发送确认信息的窗口的大小（字节）	1048576
<i>call-failover-timeout</i>	在尝试失效切换期间进行调用时使用的超时时间	-1（永不超时）
<i>notification-interval</i>	它指定当附加到群集时群集连接必须广播自己的频率（毫秒）	1,000 毫秒
<i>notification-attempts</i>	它指定当附加到群集时群集连接必须广播自己的最大次数	2
<i>discovery-group-ref</i>	这个参数指定使用哪个发现组来获取当前群集连接将连接的群集里其他节点的列表。	

在群集里节点间创建连接来组成群集连接时，HornetQ 使用在服务器配置文件（`standalone.xml` 和 `domain.xml`）里定义的一个群集用户和群集密码。

```
<cluster-user>HORNETQ.CLUSTER.ADMIN.USER</cluster-user>
<cluster-password>NEW USER</cluster-password>
```



警告

修改这些凭证的默认值来阻止远程客户使用默认值连接服务器是很重要的。

[提交 bug 报告](#)

20.12. 高可用性

20.12.1. 高可用性简介

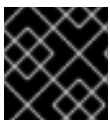
HornetQ 支持在一个或多个服务器发生故障后继续运行。这种功能部分是通过失效切换实现的，客户在在线服务器发生故障时从在线服务器移植到备份服务器。要保持备份服务器更新，消息持续地通过两个策略从在线服务器复制到备份服务器：共享存储（`shared store`）和复制（`replication`）。

有两种高可用性拓扑类型：

- **Dedicated Topology**：这个拓扑结构由两个 EAP 服务器组成。在第一个服务器里 HornetQ 被配置为在线服务器。在第二个服务器里 HornetQ 被配置为备份服务器。将 HornetQ 配置为备份服务器的 EAP 服务器，将只充当 HornetQ 容器。这个服务器是不活动的，它无法容纳部署，如 EJB、MDB 或 Servlet。
- **Collocated Topology**：这个拓扑结构包含两个 EAP 服务器。每个 EAP 服务器都包含两个 HornetQ 服务器（在线和备份服务器）。第一个 EAP 服务器上的 HornetQ 在线服务器和第二个 EAP 服务器上的 HornetQ 备份服务器组成了一个在线/备份配对。而第二个 EAP 服务器上的 HornetQ 在线服务器和第一个 EAP 服务器上的备份服务器组成了另外一个在线/备份配对。

在 **Collocated Topology** 结构里，只要在线 HornetQ 服务器（在线/配对的一部分）发生故障，备份服务器将接管并变成活动服务器。当备份 HornetQ 服务器在失效切换时关闭的情况下，备份服务器上配置的目的地和连接工厂都会从 JNDI 上取消绑定。

JNDI 是和其他在线 HornetQ 服务器（其他在线/备份配对的一部分）共享的。因此，从 JNDI 取消目的地和连接工厂的绑定也会从这个在线 HornetQ 服务器取消目的地和连接工厂的绑定。



重要

Collocated 备份服务器的配置不能包含目的地或连接工厂的配置。



注意

下面的信息引用了 `standalone-full-ha.xml`。配置的修改也适用于 `standalone-full-ha.xml`，或从中衍生的任何配置文件。

[提交 bug 报告](#)

20.12.2. 关于 HornetQ 共享存储

在使用共享存储时，在线和备份服务器都用共享文件系统共享相同的整个数据目录。这包含 **paging** 目录、**journal** 目录、大型消息和绑定日志。当失效切换发生时，备份服务器开始接管，它将从共享文件系统加载持久性存储。然后客户可以进行连接。

这种高可用性和数据复制不同，因为它要求有一个共享的文件系统供在线和备份节点访问。这通常是某种形式的高性能存储区域网络（**Storage Area Network**，**SAN**）。

共享存储高可用性的优势是在在线和备份节点间不需要进行复制。这意味着在正常操作时它不会有任何因为复制引起的性能损失。

共享存储的缺点是它要求一个共享文件系统，当备份服务器激活时，它需要从共享存储加载日志。根据存储里数据的多少，这会花费一些时间。

如果在普通操作时追求最高性能，且可以接受对快速 **SAN** 的访问及稍慢的失效切换速度（根据数据的多少），我们就会推荐高可用性的共享存储。

[提交 bug 报告](#)

20.12.3. 关于 HornetQ 存储配置

HornetQ 支持两种不同的共享存储配置：

- **SAN** 上的 **GFS2**，使用 **ASYNCIO** 日志类型。
- **NFSv4**，使用 **ASYNCIO** 或 **NIO** 日志类型。



重要

HornetQ 支持 **NFS**，但有着严格的配置准则，如下所示。

红帽企业版 **Linux NFS** 实现支持直接 **I/O**（设置 **O_DIRECT** 标记并打开文件）以及基于内核的异步 **I/O**。使用这些功能，依据严格的配置规则，你可以将 **NFS** 做为一个共享存储选项使用。

- 红帽企业版 **Linux NFS** 客户缓存必须被禁用。



注意

我们推荐如果根据上面的条件使用 **NFS**，需要使用高可用的 **NFS** 配置。

[提交 bug 报告](#)

20.12.4. 关于 HornetQ 的日志类型

HornetQ 里有两种日志类型：

- **ASYNCIO**
- **NIO**

ASYNCIO 日志类型，也称为 **AIO**，是 **Linux** 异步 **IO** 库（**AIO**）的一个瘦原生代码 **wrapper**。使用原生功

能可以提供比 NIO 更好的性能。这个日志类型只被 Red Hat 企业版 Linux 支持且需要在 JBoss EAP 6 运行的位置安装 **libaio** 和 **Native Components** 软件包。关于安装 **Native Components** 软件包的说明，请参考《安装指南》。



重要

请在 JBoss EAP 6 启动后检查服务器日志，以确保成功加载原生库且使用 **ASYNCIO** 日志类型。如果原生库加载失败，**HornetQ** 将使用 **NIO** 日志类型并在服务器日志里注明。

NIO 日志类型使用标准的 Java NIO 与文件系统交互。这提供了极佳的性能且可以运行在所有支持的平台上。

要指定 HornetQ 日志类型，请在 **Messaging** 子系统里设置 **<journal-type>** 参数。

[提交 bug 报告](#)

20.12.5. 用共享存储配置 HornetQ 的专有拓扑结构

要配置在线和备份服务器在专有拓扑结构里的共享存储，请使每台服务器的 **standalone-X.xml** 文件包含下列内容：

```
<shared-store>true</shared-store>
<paging-directory path="${shared.directory}/journal"/>
<bindings-directory path="${shared.directory}/bindings"/>
<journal-directory path="${shared.directory}/journal"/>
<large-messages-directory path="${shared.directory}/large-messages"/>
.
.
.
<cluster-connections>
  <cluster-connection name="my-cluster">
    ...
  </cluster-connection>
</cluster-connections>
```

表 20.14. HornetQ 服务器属性（在线和备份服务器）

属性	描述
shared-store	服务器是否使用共享存储。默认值为 false。
paging-directory path	指定 paging 目录的路径。在线和备份服务器共享这个目录，所以这个路径对于它们来说是相同的。
bindings-directory path	指定绑定日志的路径。在线和备份服务器共享这个日志，所以这个路径对于它们来说是相同的。
journal-directory path	指定 journal 目录的路径。在线和备份服务器共享这个目录，所以这个路径对于它们来说是相同的。
large-messages-directory path	指定大型消息目录的路径。在线和备份服务器共享这个目录，所以这个路径对于它们来说是相同的。

属性	描述
<code>failover-on-shutdown</code>	当在线或当前活动的备份服务器关闭时这个服务器是否成为活动服务器。

备份服务器也必须显性地标记为备份服务器。

```
<backup>true</backup>
```

HornetQ 备份服务器专有的设置属性是 **allow-failback**。它指定如果原来的在线服务器恢复时备份服务器是否自动关闭。

[提交 bug 报告](#)

20.12.6. HornetQ 消息复制



警告

只有持久性消息才会被复制。非持久性消息无法通过失效切换保留。

当在线服务器和备份服务器不共享相同的数据存储时，它们之间的消息复制是通过网络实现的。所有的日志都在两个服务器间进行复制，只要这两个服务器位于相同的群集里且使用相同的群集用户名和密码。在线服务器接收的所有（持久性的）数据流量都会复制到备份服务器上。

当备份服务器在线时，它会查找并连接在线服务器以试图进行同步。在它进行同步时，它无法作为备份服务器访问。取决于要同步的数据的大小及网络速度，同步可能需要较长的时间。如果备份服务器在线且没有在线服务器可用，备份服务器将等待，直至群集里的在线服务器可用。

要启用服务器复制数据，您必须在 **standalone-full-ha.xml** 文件里定义相关的一个链接。备份服务器只会用相同的组名复制在线服务器。组名必须在每台服务器的 **standalone-full-ha.xml** 文件中的 **backup-group-name** 参数里进行定义。

在在线服务器发生故障的情况下，正确配置和完全同步的备份服务器将接管其职责。备份服务器将在在线服务器发生故障而备份服务器可以连接群集里一半以上的服务器时激活。如果多过半数的服务器无法响应，它将指示网络错误，备份服务器将等待到在线服务器的连接重试。

要在失效切换后恢复到原始状态，您有必要启动在线服务器并等待它和备份服务器完全同步。当这一步完成后，您可以关闭备份服务器以让原始的在线服务器再次激活。如果设置 **allow-failback** 属性为 **true**，这一步会自动完成。

[提交 bug 报告](#)

20.12.7. 配置 HornetQ 服务器的复制

要配置在线和备份服务器为复制配对，请使两者的 **standalone-full-ha.xml** 文件里有如下设置：

```

<shared-store>false</shared-store>
<backup-group-name>NameOfLiveBackupPair</backup-group-name>
<check-for-live-server>true</check-for-live-server>
.
.
.
<cluster-connections>
  <cluster-connection name="my-cluster">
    ...
  </cluster-connection>
</cluster-connections>

```

表 20.15. HornetQ 复制设置属性

属性	描述
shared-store	服务器是否使用共享存储。默认值为 false。
backup-group-name	标识应该彼此复制的在线/备份服务器配对的唯一名称
check-for-live-server	被复制的在线服务器是否必须检查当前的群集来查看是否已有具有相同节点 ID 的在线服务器。默认值为 false。
failover-on-shutdown	在普通的服务器关闭时，这个备份服务器（如果它是备份服务器）是变成在线服务器。默认值为 false。

备份服务器也必须显性地标记。

```
<backup>true</backup>
```

表 20.16. HornetQ 备份服务器的设置属性

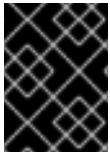
属性	描述
allow-failback	如果原来的在线服务器重新上线，是否自动关闭这个服务器。它的默认值为 true。
max-saved-replicated-journal-size	在失效切换发生后备份日志的最大数目。只有 allow-failback 为 true 时才有必要指定这个属性。默认值为 2，表示两次失效切换后，备份服务器必须重启才能够从在线服务器复制日志并再次成为备份服务器。

[提交 bug 报告](#)

20.12.8. 关于高可用性（HA）失效切换

高可用性失效切换分为自动客户失效切换和使用在线-备份结构的应用程序级别的失效切换。每个在线服务器都有一个备份服务器。目前只支持每个在线服务器对应一个备份服务器。

只有在线服务器崩溃时备份服务器才会接管。在在线服务器重启后，如果 **allow-failback** 属性被设置为 **true**，它会再次称为在线服务器。当原来的在线服务器接管后，备份服务器将恢复成在线服务器的备份服务器。



重要

即使您没有使用群集功能，群集也应该启用。这是因为 HA 群集的每个节点都必须有一个到其他节点的群集连接，从而能够和其他服务器协商角色。

在线和备份服务器实现了高可用性群集拓扑结构，它们使用 IP 多点传送发送连接细节。如果无法使用 IP 多点传送，也可能使用初始连接的静态配置。在初始连接之后，系统会将这个拓扑结构通知客户。如果当前的连接是稳定的，客户将和另外一个节点建立新的连接。

在在线服务器崩溃且备份服务器接管后，您需要重启在线服务器并将客户进行失效切换。为此，重启原来的在线服务器并终止新的在线服务器。您可以终止服务器进程或等待服务器自己崩溃。在 **standalone.xml** 配置文件里设置 **failover-on-shutdown** 为 **true**，您也可以让服务器在正常关闭时进行失效切换。

```
<failover-on-shutdown>true</failover-on-shutdown>
```

在默认情况下，**failover-on-shutdown** 属性为 **false**。

通过在 **standalone.xml** 配置文件里设置 **allow-failback** 属性为 **true**，您也可以在旧的在线服务器恢复时自动进行接管且强制关闭新的在线服务器。

```
<allow-failback>true</allow-failback>
```

在 Replication HA 模式下，如果在 **standalone.xml** 配置文件里设置 **check-for-live-server** 属性为 **true**，当旧的在线服务器恢复时，强制新的在线服务器关闭。

```
<check-for-live-server>true</check-for-live-server>
```

[提交 bug 报告](#)

20.12.9. HornetQ 备份服务器上的部署

在指定的 HA 环境里，将 HornetQ 配置为备份服务器的 JBoss EAP 6 服务器不能用于运行任何使用或连接 HornetQ 备份服务器的部署。这包括 EJB（Stateless Session Beans、Message Driven Beans）或 Servlet。

如果 JBoss EAP 6 服务器具有 HornetQ 备份配置（消息子系统里配置为 'live' 而另外一个 HornetQ 服务器配置为 'backup'），JBoss EAP 6 服务器可以运行部署，只要这些部署是配置程连接 'live' HornetQ 服务器的。

[提交 bug 报告](#)

第 21 章 事务子系统

21.1. 事务子系统的配置

21.1.1. 事务配置概述

介绍

下面的过程展示了如何配置 JBoss EAP 6 的事务子系统。

- [第 21.1.3 节 “用 JTA 事务配置您的数据源”](#)
- [第 21.1.4 节 “配置 XA 数据源”](#)
- [第 21.1.2 节 “配置事务管理者”](#)
- [第 21.1.6 节 “为事务子系统配置日志”](#)

[提交 bug 报告](#)

21.1.2. 配置事务管理者

您可以使用基于 Web 的管理控制台或命令行管理 CLI 来配置事务管理者（Transaction Manager，TM）。对于每条命令或选项，我们都假设您在受管域里运行 JBoss EAP 6。如果您使用的是独立服务器或者您想使用 **default** 之外的配置集，您可能需要修改下面的步骤和命令。

命令示例注记

- 在管理控制台里，第一次登录时会自动选择 **default** 配置集。如果您想改动到其他配置集，请选择其他配置集。

类似地，请在 CLI 命令示例里替换 **default** 配置集。

- 如果您使用的是独立服务器，则只存在一个配置集。请忽略选择配置集的说明。在 CLI 命令示例里，请删除 **/profile=default** 部分。



注意

为了使 TM 选项在管理控制台或管理 CLI 里可见，您必须启用 **transactions**。它默认是启用的，对于许多其他子系统的正常运行，它也是必需的，所以不太可能禁用它。

用管理控制台配置事务管理者

要使用基于 Web 的管理控制台配置 TM，请从管理控制台屏幕的顶部选择 **Configuration** 标签页。如果您使用受管域，您可以在左上角的 **Profile** 选择框里选择正确的配置集。展开 **Container** 菜单并选择 **Transactions**。

事务管理者配置页面里显示了多数的选项。**Recovery** 选项默认是隐藏的。点击 **Recovery** 标签页来查看恢复选项。然后点击 **Edit** 按钮来编辑任何选项。修改会立即生效。

点击 **Need Help?** 标签显示在线帮助文本。

用管理 CLI 配置事务管理者

在管理 CLI 里，您可以使用一系列命令来配置事务管理者。对于使用 **default** 的受管域，这些命令都以 **/profile=default/subsystem=transactions/** 开始，而对于独立服务器则使用 **/subsystem=transactions**。



重要

HornetQ 不允许多个实例共享消息日志库。如果您配置了多个 HornetQ 实例，每个实例都必须有自己地消息日志库。

表 21.1. 事务管理者配置选项

选项	描述	CLI 命令
Enable Statistics	是否启用事务统计。这些统计可以在管理控制台的 Runtime 标签页里的 Subsystem Metrics 部分查看。	<code>/profile=default/subsystem=transactions/:write-attribute(name=enable-statistics,value=true)</code>
Default Timeout	默认的事务超时时间。默认值是 300 秒。您可以在程序里对每个事务覆盖这个值。	<code>/profile=default/subsystem=transactions/:write-attribute(name=default-timeout,value=300)</code>
Object Store Path	TM 对象库存储数据的相对或绝对文件系统路径。默认是相对于 object-store-relative-to 参数的路径。	<code>/profile=default/subsystem=transactions/:write-attribute(name=object-store-path,value=tx-object-store)</code>
Object Store Path Relative To	引用域模型里的全局路径配置。默认值是 JBoss EAP 6 的数据目录，即 jboss.server.data.dir 属性的值。对于受管域，默认是 EAP_HOME/domain/data/ ，而对于独立服务器是 EAP_HOME/standalone/data/ 。对象库的 object-store-path TM 属性是相对于这个路径的值。	<code>/profile=default/subsystem=transactions/:write-attribute(name=object-store-relative-to,value=jboss.server.data.dir)</code>
Socket Binding	当使用基于套接字的机制时，指定事务管理者进行恢复和生成事务标识符时使用的套接字绑定的名称。关于生成唯一标识符的更多信息，请参考 process-id-socket-max-ports 。您可以在管理控制台的 Server 标签页上为每个服务器组指定套接字绑定。	<code>/profile=default/subsystem=transactions/:write-attribute(name=socket-binding,value=txn-recovery-environment)</code>
Recovery Listener	事务恢复进程是否应该侦听网络接口。默认值为 false 。	<code>/profile=default/subsystem=transactions/:write-attribute(name=recovery-listener,value=false)</code>

下面是高级选项，只能用管理 CLI 进行修改。根据默认配置进行修改时请小心行事。更多信息请联系红帽全球支持服务。

表 21.2. 高级 TM 配置选项

选项	描述	CLI 命令
jts	是否使用 Java Transaction Service (JTS) 事务。默认值为 false ，表示只使用 JTA 事务。	<code>/profile=default/subsystem=transactions/:write - attribute(name=jts, value=false)</code>
node-identifier	<p>事务管理者的节点标识符。这个选项在下列情况下是必需的：</p> <ul style="list-style-type: none"> • JTS 和 JTS 间的通讯 • 当两个事务管理者访问共享的资源管理者 • 当两个事务管理者访问共享的对象库 <p>对于每个事务管理者来说，node-identifier 都必须是唯一的，因为它要求在恢复期间强制数据的一致性。对于 JTA 来说 node-identifier 也必须是唯一的，因为多个节点可能和系统的资源管理者交互或分享事务对象库。</p>	<code>/profile=default/subsystem=transactions/:write - attribute(name=node-identifier, value=1)</code>
process-id-socket-max-ports	<p>事务管理者为每个事务日志创建唯一的标识符。为生成唯一标识符提供两个不同的机制：基于套接字和基于进程标识符。</p> <p>对于基于套接字的标识符，套接字被打开，其端口号用标识符。如果端口已在使用，将探测下一个端口，直到找到空闲的端口。process-id-socket-max-ports 代表 TM 在失败前将尝试的最多套接字数量。默认值是 10。</p>	<code>/profile=default/subsystem=transactions/:write - attribute(name=process-id-socket-max-ports, value=10)</code>
process-id-uuid	true 表示使用进程标识符来为每个事务创建唯一的标识符。否则将使用基于套接字的机制。默认值为 true 。更多信息请参考 process-id-socket-max-ports 。	<code>/profile=default/subsystem=transactions/:write - attribute(name=process-id-uuid, value=true)</code>

选项	描述	CLI 命令
use-hornetq-store	对于事务日志，使用 HornetQ 的日志存储机制而不是基于文件的存储。这默认是禁用的，但可能会提高 I/O 性能。我们不推荐对 JTS 事务使用独立的事务管理者。当修改这个选项时，服务器必须用 shutdown 重启以使修改生效。	<code>/profile=default/subsystem=transactions/:write-attribute(name=use-hornetq-store,value=false)</code>

[提交 bug 报告](#)

21.1.3. 用 JTA 事务配置您的数据源

概述

本节展示了如何启用数据源上的 JTA。

前提条件

在开始安装前，你必须满足下列条件：

- 您的数据库或其他资源必须支持 JTA。如果有疑问，请参考数据库或其他资源的相关文档。
- 创建数据库。请参考 [第 6.3.1 节“用管理界面创建一个 Non-XA 数据源”](#)。
- 停止 JBoss EAP 6 服务器。
- 在文本编辑器里编辑服务器配置文件。

过程 21.1. 配置数据源以使用 JTA 事务

1. 在文本编辑器里打开配置文件。

根据您是否以受管域还是独立服务器运行 JBoss EAP 6，您的配置文件所处的位置不同。

◦ 受管域

受管域的默认配置文件对于 Red Hat 企业版来说，位于 **EAP_HOME/domain/configuration/domain.xml**，对于 Microsoft Windows 则位于 **EAP_HOME\domain\configuration\domain.xml**。

◦ 独立服务器

独立服务器的默认配置文件对于 Red Hat 企业版 Linux 来说，位于 **EAP_HOME/standalone/configuration/standalone.xml**，对于 Microsoft Windows，位于 **EAP_HOME\standalone\configuration\standalone.xml**。

2. 定位对应数据源的 <datasource> 标签。

数据源将 **jndi-name** 属性设置为您创建时指定的值。例如，ExampleDS 数据源应类似于：

```
<datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="H2DS" enabled="true" jta="true" use-java-context="true" use-ccm="true">
```

3. 设置 jta 属性为 true。

和之前的步骤一样，添加下列内容到 <datasource> 标记里：**jta="true"**。

- 4. 保存配置文件。
保存配置文件并退出文本编辑器。
- 5. 启动 JBoss EAP 6。
重新启动 JBoss EAP 6 服务器。

结果：
JBoss EAP 6 已启动且配置您的数据源使用 JTA 事务。

[提交 bug 报告](#)

21.1.4. 配置 XA 数据源

前提条件

为了添加 XA 数据源，你需要登录至管理控制台。详情请参考 [第 3.4.2 节“登录到管理控制台”](#)。

- 1. 添加新的数据源。
添加新的数据源到 JBoss EAP 6。请按照 [第 6.3.1 节“用管理界面创建一个 Non-XA 数据源”](#) 里的步骤，但要点击顶部的 **XA Datasource** 标签。
- 2. 配置其他属性。
[第 6.6.1 节“数据源参数”](#) 列出了所有的数据源参数。

结果
配置好的 XA 数据源可以使用了。

[提交 bug 报告](#)

21.1.5. 关于事务日志消息

要跟踪事务状态同时也使日志文件可读，对事务 logger 请使用 **DEBUG** 日志级别。详细地调试信息则使用 **TRACE** 日志级别。关于配置事务 logger 的信息，请参考 [第 21.1.6 节“为事务子系统配置日志”](#)。

当配置为 **TRACE** 日志级别时，事务管理者可以生成大量的日志信息。下面是一些最常见的信息。这个列表并不完整，所以实际环境里您可能会看到其他信息。

表 21.3. 事务状态变化

事务开始	<div>当事务开始时，下列代码将被执行：</div> <pre>com.arjuna.ats.arjuna.coordinator .BasicAction::Begin:1342 tsLogger.logger.trace("BasicActio n::Begin() for action-id "+ get_uid());</pre>
------	--

事务提交	<p>当事务提交时，下列代码将被执行：</p> <pre>com.arjuna.ats.arjuna.coordinator .BasicAction::End:1342 tsLogger.logger.trace("BasicActio n::End() for action-id "+ get_uid());</pre>
事务回滚	<p>当事务回滚时，下列代码将被执行：</p> <pre>com.arjuna.ats.arjuna.coordinator .BasicAction::Abort:1575 tsLogger.logger.trace("BasicActio n::Abort() for action-id "+ get_uid());</pre>
事务超时	<p>当事务超时时，下列代码将被执行：</p> <pre>com.arjuna.ats.arjuna.coordinator .TransactionReaper::doCancellatio ns:349 tsLogger.logger.trace("Reaper Worker " + Thread.currentThread() + " attempting to cancel " + e._control.get_uid());</pre> <p>您将看到相同的线程回滚上面显示的事务。</p>

[提交 bug 报告](#)

21.1.6. 为事务子系统配置日志

概述

使用这个过程来控制事务日志信息的数量，它独立于 JBoss EAP 6 里的其他日志设置。主过程展示了如何在基于 Web 的管理控制台里实现这一点。随后给出了管理 CLI 命令。

过程 21.2. 用管理控制台配置事务 Logger

1. 进入日志配置区域。
在管理控制台里，点击 **Configuration** 标签页。如果您在使用受管域，请在右上角的 **Profile** 选择框里选择要配置的服务器配置集。

展开 **Core** 菜单，选择 **Logging**。
2. 编辑 `com.arjuna` 属性。

选择 **Log Categories** 标签页。选择 **com.arjuna** 并点击 **Details** 部分的 **Edit** 按钮。在这里您可以添加类专有的日志信息。**com.arjuna** 类是现有的。您可以修改日志级别以及是否使用父处理程序。

日志级别

默认的日志级别是 **WARN**。因为事务可以产生大量的日志输出，标准日志级别对于事务 **logger** 来说有稍许不同。通常比所选日志级别严重性较低的消息会被舍弃。

事务日志级别，从最精简到最冗余。

- TRACE
- DEBUG
- INFO
- WARN
- ERROR
- FAILURE

使用父处理程序

Logger 是否应该将输出发送到父 **Logger**。默认行为是 **true**。

3. 修改会立即生效。

[提交 bug 报告](#)

21.2. 事务管理

21.2.1. 浏览并管理事务

管理 CLI 有浏览和操纵事务记录的功能。这个功能是通过事务管理者和 JBoss EAP 6 的管理 API 交互作用来提供的。

事务管理者将每个待定事务和涉及的参与者的信息存储在一个名为**对象库 (object store)** 的持久性存储里。管理 API 将对象库作为名为 **log-store** 的资源开放。名为 **probe** 的 API 操作读取事务日志并为每个日志创建一个节点。每当您需要刷新 **log-store** 时，您都可以手动调用 **probe** 命令。事务日志快速出现和消失都是正常的。

例 21.1. 刷新日志库

这个命令刷新了在受管域里使用配置集 **default** 的服务器组的日志库。对于独立服务器，请从命令里删除 **profile=default**。

```
/profile=default/subsystem=transactions/log-store=log-store/:probe
```

例 21.2. 查看所有 Prepared 事务

要查看所有 **Prepared** 事务，首先请刷新日志库（请参考 [例 21.1 “刷新日志库”](#)），然后运行下列命令（功能类似于文件系统的 **ls** 命令）。

```
ls /profile=default/subsystem=transactions/log-store=log-store/transactions
```

这会展示每个事务及其唯一的标识符。可对单独的事务执行单独的操作（请参考 [管理事务](#)）。

管理事务

查看事务的属性。

要查看事务的信息，如 JNDI 名称、EIS 产品名和版本或状态，请使用 **:read-resource** CLI 命令。

```
/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:read-resource
```

查看事务的参与者。

每个事务日志都包含一个名为 **participants** 的子元素。对这个元素使用 **read-resource** CLI 命令来查看事务的参与者。参与者是通过它们的 JNDI 名称来标识的。

```
/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9/participants=java\:/JmsXA:read-resource
```

结果可能类似于：

```
{
  "outcome" => "success",
  "result" => {
    "eis-product-name" => "HornetQ",
    "eis-product-version" => "2.0",
    "jndi-name" => "java:/JmsXA",
    "status" => "HEURISTIC",
    "type" => "/StateManager/AbstractRecord/XAResourceRecord"
  }
}
```

这里显示的结果状态是 **HEURISTIC** 且符合恢复的条件。详情请参考 [恢复事务](#)。。

删除事务。

每个事务日志都支持 **:delete** 操作来删除代表事务的事务日志。

```
/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:delete
```

恢复事务。

每个事务都支持通过 **:recover** CLI 命令进行恢复。

恢复启发式事务和参与者

- 如果事务的状态是 **HEURISTIC**，恢复操作会修改其状态为 **PREPARE** 并触发恢复。
- 如果事务的其中一个参与者是启发式的，恢复操作会试图重新执行 **commit** 操作。如果成功，参与者将从事务日志里删除。您可以通过在 **log-store** 上重新运行 **:probe** 并检查参与者是否不再列出来验证这一点。如果这是最后一个参与者，事务也会被删除。

刷新需要恢复的事务的状态。

如果事务需要恢复，在进行恢复之前，您可以使用 **:refresh** CLI 命令确定它仍要求恢复。

```
/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:refresh
```

查看事务子系统的统计信息

如果启用了事务管理者（Transaction Manager，TM）统计，您可以查看事务管理者和事务子系统的统计信息。关于如何启用 TM 统计的信息，请参考 第 21.1.2 节“配置事务管理者”。

您可以用管理控制台或管理 CLI 查看统计信息。在管理控制台里，你可以通过 **Runtime → Status → Subsystems → Transactions** 进行查看。对于受管域里的每台服务器，事务统计信息都是是可用的。要查看不同服务器的状态，请在左侧的菜单里选择 **Change Server**，然后从列表里选择服务器。

下表展示了每个可用的统计信息、描述和查看统计信息的 CLI 命令。

表 21.4. 事务子系统的统计信息

统计内容	描述	CLI 命令
Total	服务器上事务管理者处理的事务总数。	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-transactions,include-defaults=true)</pre>
Committed	服务器上事务管理者提交的事务总数。	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-committed-transactions,include-defaults=true)</pre>

统计内容	描述	CLI 命令
Aborted	服务器上事务管理者中止的事务总数。	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-aborted-transactions,include-defaults=true)</pre>
Timed Out	服务器上事务管理者处理的超时事务总数。	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-timed-out-transactions,include-defaults=true)</pre>
Heuristics	在管理控制台里不可用。处于 heuristic 状态的事务的数量。	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-heuristics,include-defaults=true)</pre>
In-Flight Transactions	在管理控制台里不可用。已经开始但还未终止的事务的数量。	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-inflight-transactions,include-defaults=true)</pre>

统计内容	描述	CLI 命令
Failure Origin - Applications	故障来源是应用程序的失败事务的数量。	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-application-rollback,include-defaults=true)</pre>
Failure Origin - Resources	故障来源是资源的失败事务的数量。	<pre>/host=master/server=server-one/subsystem=transactions/:read-attribute(name=number-of-resource-rollback,include-defaults=true)</pre>

[提交 bug 报告](#)

21.3. 事务引用

21.3.1. JBoss 事务的错误和异常

关于 `UserTransaction` 类的方法抛出的异常，请参考 <http://download.oracle.com/javaee/1.3/api/javax/transaction/UserTransaction.html> 上的 `UserTransaction API` 规格。

[提交 bug 报告](#)

21.3.2. JTA 事务的限制

JTA 事务不能跨多个 JBoss EAP 实例间进行分布。你可以使用 JTS 事务来实现这一点。

要使用 JTS 事务，您需要配置 ORB，它包括在 JacORB 子系统里启用事务，然后配置 JTS 子系统。

- [第 21.4.2 节 “为 JTS 事务配置 ORB”](#)

[提交 bug 报告](#)

21.4. ORB 配置

21.4.1. 关于公共对象请求代理体系结构（Common Object Request Broker Architecture，CORBA）

Common Object Request Broker Architecture (CORBA) 是一个启用应用程序和服务使其一起运行的标准，

即使这些程序或服务是用多种不兼容的语言编写或运行在独立的平台上的。**CORBA** 请求由一个名为 **Object Request Broker (ORB)** 的服务器端的组件进行代理。**JBoss EAP 6** 提供一个 **ORB** 实例来作为 **JacORB** 组件。

这个 **ORB** 用于 **Java Transaction Service (JTS)** 事务内部，也可以供您自己的应用程序使用。

[提交 bug 报告](#)

21.4.2. 为 JTS 事务配置 ORB

在默认的 **JBoss EAP 6** 的安装里，**ORB** 是禁用的。你可以用命令行管理 **CLI** 启用 **ORB**。



注意

在受管域里，**JacORB** 子系统只能用于 **full** 和 **full-ha** 配置集。在独立服务器里，你可以使用 **standalone-full.xml** 或 **standalone-full-ha.xml** 配置。

过程 21.3. 使用管理控制台配置 ORB

1. 查看配置集设置。

从管理控制台的顶部选择 **Configuration**。如果你使用了受管域，请在左上角选择 **full** 或 **full-ha** 配置集。

2. 修改 Initializers 设置

展开 **Subsystems** 菜单，展开 **Container** 子菜单并选择 **JacORB**。

在主屏幕上出现的表单里，选择 **Initializers** 标签页并点击 **Edit** 按钮。

通过设置 **Security** 为 **on** 来启用安全拦截器。

要启用 **JTS** 里的 **ORB**，请设置 **Transaction Interceptors** 值为 **on**，而不是默认的 **spec**。

关于这些值的详细解释，请点击表单里的 **Need Help?** 链接。在完成编辑后请点击 **Save**。

3. 高级的 ORB 配置

关于高级的配置选项，请参考表单的其他部分。每个部分都包含一个关于参数详细解释的 **Need Help?** 链接。

使用管理 CLI 配置 ORB

你可以使用管理 **CLI** 配置 **ORB** 的每个方面。下面的命令配置初始器为与上面过程里使用管理控制台相同的值。这是 **JTS** 里 **ORB** 的最小配置。

这些命令是为使用 **full** 配置集的受管域配置的。如果有必要，请根据需要修改这个配置集。如果你使用了独立服务器，请忽略命令行的 **/profile=full** 部分。

例 21.3. 启用安全拦截器

```
/profile=full/subsystem=jacorb/:write-attribute(name=security,value=on)
```

例 21.4. 在 JacORB 子系统里启用事务

```
/profile=full/subsystem=jacorb/:write-
```

```
attribute(name=transactions,value=on)
```

例 21.5. 在事务子系统里启用 JTS

```
/profile=full/subsystem=transactions:write-  
attribute(name=jts,value=true)
```



注意

对于 JTS 激活，服务器必须重启，重载是不够的。

[提交 bug 报告](#)

21.5. JDBC 对象库的支持

21.5.1. 事务的 JDBC 库

前提条件：

- [第 3.5.4 节 “用管理 CLI 连接受管服务器实例”](#)

事务可将 JDBC 数据源用作其对象库。如果要使用的数据库是为失效切换和恢复配置的，相比使用应用程序上的磁盘空间这是更好的选择。但需要权衡的是，原始的 JDBC 对象库是特殊的对象库，性能可能没有文件系统或 HornetQ 日志对象库那样好。



注意

用作 Transactions 对象的 JDBC 数据源必须在服务器的配置文件里指定 `jta="false"`。

过程 21.4. 将 JDBC 数据源启用为 Transactions 对象库

1. 将 `use-jdbc-store` 设置为 `true`。

```
/subsystem=transactions:write-attribute(name=use-jdbc-store,  
value=true)
```

2. 设置 `jdbc-store-datasource` 为数据源要使用的 JNDI 名称。

```
/subsystem=transactions:write-attribute(name=jdbc-store-datasource,  
value=java:jboss/datasources/TransDS)
```

3. 重启 JBoss EAP 6 服务器以使修改生效。

```
shutdown --restart=true
```

下面是完整的属性列表。

表 21.5. 事务 JDBC 库属性

属性	描述
use-jdbc-store	设置为 "true" 为事务启用 JDBC 库。
jdbc-store-datasource	用于存储的 JDBC 数据源的 JNDI 名称。
jdbc-action-store-drop-table	在启动时丢弃并重新创建 Action Store 表。可选项，默认值为 "False"。
jdbc-action-store-table-prefix	Action Store 表名的前缀。可选项。
jdbc-communication-store-drop-table	在启动时丢弃并重新创建 Communication Store 表。可选项，默认值为 "False"。
jdbc-communication-store-table-prefix	Communication Store 表名的前缀。可选项。
jdbc-state-store-drop-table	在启动时丢弃并重新创建 State Store 表。可选项，默认值为 "False"。
jdbc-state-store-table-prefix	State Store 表名的前缀。可选项。

还可查看：

- [第 21.1.3 节 “用 JTA 事务配置您的数据源”](#)

[提交 bug 报告](#)

第 22 章 邮件子系统

22.1. 在邮件子系统里使用自定义传输

当使用标准的邮件服务器（POP3、IMAP）时，服务器可以定义一系列属性，其中一些是必需的。

最重要的属性是 **outbound-socket-binding-ref**，它是对转出邮件套接字绑定的引用，且是用主机地址和端口号码来定义的。

对于某些用户来说，这并非最有效的解决方案，因为他们的主机配置将多个主机用于负载平衡。然而，要求用户实现自定义邮件传输的标准的 JavaMail 不支持这个配置。

这些自定义的传输不要求 **outbound-socket-binding-ref** 且允许自定义的主机属性格式。

自定义传输可以通过下列 CLI 命令来进行配置：

过程 22.1.

1. 添加新的邮件会话。下面的命令创建名为 **mySession** 的新会话并设置 JNDI 为 **java:jboss/mail/MySession**：

```
/subsystem=mail/mail-session=mySession:add(jndi-name=java:jboss/mail/MySession)
```

2. 添加转出套接字绑定。下面的命令添加了名为 **my-smtp-binding** 的套接字绑定，它指向 **localhost:25**。

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-smtp-binding:add(host=localhost, port=25)
```

3. 用 **outbind-socket-binding-ref** 添加 SMTP 服务器。下面的命令添加一个名为 **my-smtp-binding** 的 SMTP，并定义一个用户名、密码和 TLS 配置。

```
/subsystem=mail/mail-session=mySession/server=smtp:add(outbound-socket-binding-ref= my-smtp-binding, username=user, password=pass, tls=true)
```

4. 对 POP3 和 IMAP 重复这个过程：

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-pop3-binding:add(host=localhost, port=110)
```

```
/subsystem=mail/mail-session=mySession/server=pop3:add(outbound-socket-binding-ref=my-pop3-binding, username=user, password=pass)
```

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-imap-binding:add(host=localhost, port=143)
```

```
/subsystem=mail/mail-session=mySession/server=imap:add(outbound-socket-binding-ref=my-imap-binding, username=user, password=pass)
```

5. 要使用自定义服务器，创建一个新的不带有转出套接字绑定（因为是可选项）的自定义邮件服务器，并将主机信息作为属性的一部分来提供。

```
/subsystem=mail/mail-
session=mySession/custom=myCustomServer:add(username=user,password=p
ass, properties={"host" => "myhost", "my-property" =>"value"})
```

在定义自定义协议时，任何包含句点 (.) 的属性名都被当作全限定名且直接传入。任何其他格式（如 *my-property*）将转换程下列格式：**mail.server-name.my-property**。

下面是一个完整的 XML 配置示例，它在 **custom-server** 属性里强调了自定义格式：

```
<subsystem xmlns="urn:jboss:domain:mail:1.1">
  <mail-session jndi-name="java:/Mail" from="user.name@domain.org">
    <smtp-server outbound-socket-binding-ref="mail-smtp" tls="true">
      <login name="user" password="password"/>
    </smtp-server>
    <pop3-server outbound-socket-binding-ref="mail-pop3"/>
    <imap-server outbound-socket-binding-ref="mail-imap">
      <login name="nobody" password="password"/>
    </imap-server>
  </mail-session>
  <mail-session debug="true" jndi-name="java:jboss/mail/Default">
    <smtp-server outbound-socket-binding-ref="mail-smtp"/>
  </mail-session>
  <mail-session debug="true" jndi-name="java:jboss/mail/Custom">
    <custom-server name="smtp">
      <login name="username" password="password"/>
      <property name="host" value="mail.example.com"/>
    </custom-server>
    <custom-server name="pop3" outbound-socket-binding-ref="mail-
pop3">
      <property name="custom_prop" value="some-custom-prop-value"/>
      <property name="some.fully.qualified.property" value="fully-
qualified-prop-name"/>
    </custom-server>
  </mail-session>
  <mail-session debug="true" jndi-name="java:jboss/mail/Custom2">
    <custom-server name="pop3" outbound-socket-binding-ref="mail-
pop3">
      <property name="custom_prop" value="some-custom-prop-value"/>
    </custom-server>
  </mail-session>
</subsystem>
```

[提交 bug 报告](#)

第 23 章 EJB

23.1. 介绍

23.1.1. EJB 概述

EJB 3.1 是开发分布式、事务性、安全和可移植 Java EE 应用程序的 API，它使用了名为 Enterprise Bean 的服务器端组件。EJB 以鼓励代码重用的松耦合方式实现了应用程序的商业逻辑。EJB 3.1 是作为 Java EE 规格 JSR-318 编写文档的。

JBoss EAP 6 对使用 EJB 3.1 规格构建的应用程序提供了完整的支持。

[提交 bug 报告](#)

23.1.2. 用于管理员的 EJB 概述

JBoss 管理员可以通过许多配置选项来控制 JBoss EAP 6 里的 EJB 的性能。这些选项可以通过管理控制台或命令行配置工具来访问。编辑 XML 服务器配置文件来应用修改也是可能的，但我们不推荐这么做。

根据服务器运行的方式，EJB 配置选项在管理控制台里所处的位置会有轻微的不同。

1. 点击管理控制台顶部的 **Configuration** 标签。
2. 如果服务器运行于域模式，请从左上角的 **Profile** 下拉菜单里选择合适的配置集。
3. 展开 **Subsystems** 菜单。
4. 展开 **Container** 菜单，然后选择 **EJB 3**。

[提交 bug 报告](#)

23.1.3. Enterprise Bean

如 EJB 3.1 规格 JSR-318 里所定义的，Enterprise bean 是服务器端的组件。Enterprise bean 的目的是以分离的方式实现应用程序商业逻辑，从而鼓励重用。

EJB 以 Java 类编写且用合适的 EJB 注解进行注解。它们可以部署到自己归档（JAR 文件）里的应用服务器，也可以作为 Java EE 应用程序的一部分进行部署。应用服务器管理每个 EJB 的生命周期并提供服务，如安全性、事务和并行管理。

EJB 也可以定义任意数量的商业接口。商业接口提供对 bean 的哪些方法可被客户使用的更多控制，且允许访问运行在远程 JVM 里的客户。

有三种类型的 EJB：Session beans、Message-driven beans 和 Entity beans。



重要

EJB 3.1 里现在已废弃了 Entity Bean，Red Hat 推荐使用 JPA 实体来替代。Red Hat 只推荐出于对以前的系统的向后兼容性才使用 Entity Bean。

[提交 bug 报告](#)

23.1.4. Session Beans

Session Bean 是封装了相关商业过程或任务的 EJB，它被注入到请求它们的类里。有三种 Session Bean：stateless、stateful 和 singleton。

[提交 bug 报告](#)

23.1.5. Message-Driven Bean

Message-driven Bean(MDB) 为应用程序的开发提供了一个事件驱动的模式。MDB 的方法不会注入或从客户代码里调用，但会被消息服务（如 JMS 服务器）里的消息触发。Java EE 6 规格要求只支持 JMS，但也可以支持其他的消息系统。

[提交 bug 报告](#)

23.2. 配置 BEAN POOLS

23.2.1. Bean 池

JBoss EAP 6 在内存里维护着大量的已部署的 stateless EJB 以提供更好的性能。这个技术被称为 Bean Pooling。当 Bean 被请求时，应用服务器可以从池里取出已可用的合适的 bean 而不是创建一个新的。当 bean 不再被需要时，它将被返回到池里以供重用。

对于 stateless session bean 和 message-driven bean，Bean 池是单独配置和维护的。

[提交 bug 报告](#)

23.2.2. 创建 Bean 池

Bean 池可以用管理控制台和 CLI 工具来创建。

您可以通过文本编辑器添加所需的 Bean 池配置到服务器配置文件来创建 Bean 池。[例 23.2 “XML 配置示例”](#) 是一个配置文件的例子。

过程 23.1. 用管理控制台来创建 Bean 池

1. 登录到管理控制台。请参考 [第 3.4.2 节“登录到管理控制台”](#)。
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Bean Pools** 标签页。
3. 点击 **Add** 按钮。**Add EJB3 Bean Pools** 对话框将会出现。
4. 指定所需的细节，包括：**Name**、**Max Pool Size**、**Timeout** 和 **Timeout**。
5. 点击 **Save** 按钮完成。

过程 23.2. 用 CLI 创建 Bean 池

1. 登录 CLI 工具并连接到您的服务器。请参考 [第 3.5.4 节“用管理 CLI 连接受管服务器实例”](#)。
2. 使用 **add** 操作和下列语法。

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:add(max-pool-size=MAXSIZE, timeout=TIMEOUT, timeout-unit="UNIT")
```


- 用 Bean 池的名称替换 *BEANPOOLNAME*。
- 用 Bean 池的最大大小替换 *MAXSIZE*。
- 替代 *TIMEOUT*
- 用所需的时间单元替代 *UNIT*。所允许的值有：*NANOSECONDS*、*MICROSECONDS*、*MILLISECONDS*、*SECONDS*、*MINUTES*、*HOURS* 和 *DAYS*。

3. 请用 **read-resource** 操作来确认 Bean 池的创建。

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:read-resource
```

例 23.1. 用 CLI 创建 Bean 池

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-pool=ACCTS_BEAN_POOL:add(max-pool-size=500, timeout=5000, timeout-unit="SECONDS")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

例 23.2. XML 配置示例

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">

  <pools>

    <bean-instance-pools>

      <strict-max-pool name="slsb-strict-max-pool" max-pool-size="20"
        instance-acquisition-timeout="5"
        instance-acquisition-timeout-unit="MINUTES" />

      <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20"
        instance-acquisition-timeout="5"
        instance-acquisition-timeout-unit="MINUTES" />

    </bean-instance-pools>

  </pools>

</subsystem>
```

[提交 bug 报告](#)

23.2.3. 删除 Bean 池

未使用的 Bean 池可以用管理控制台进行删除。

预备条件：

- 您要删除的 Bean 池不能正在被使用。请参考 [第 23.2.5 节“为 Session 和 Message-Driven Bean 分配 Bean 池”](#) 以确保它没有被使用。

过程 23.3. 用管理控制台来删除 Bean 池

1. 登录到管理控制台。请参考 [第 3.4.2 节“登录到管理控制台”](#)。
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Bean Pools** 标签页。
3. 从列表里选择要删除的 Bean 池。
4. 点击 **Remove** 按钮。**Remove Item** 对话框将会出现。
5. 点击 **Confirm** 确认。

过程 23.4. 用 CLI 删除 Bean 池

1. 登录 CLI 工具并连接到您的服务器。请参考 [第 3.5.4 节“用管理 CLI 连接受管服务器实例”](#)。
2. 使用 **remove** 操作和下列语法。

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:remove
```

- 用 Bean 池的名称替换 *BEANPOOLNAME*。

例 23.3. 用 CLI 删除 Bean 池

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-  
pool=ACCTS_BEAN_POOL:remove  
{ "outcome" => "success" }  
[standalone@localhost:9999 /]
```

[提交 bug 报告](#)

23.2.4. 编辑 Bean 池

Bean 池可以用管理控制台进行编辑。

过程 23.5. 用管理控制台编辑 Bean 池

1. 登录到管理控制台。[第 3.4.2 节“登录到管理控制台”](#)
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Bean Pools** 标签页。
3. 选择要编辑的 Bean 池。
4. 点击 **Edit**。
5. 编辑要修改的内容，您只可以修改 **Max Pool Size**、**Timeout** 和 **Timeout Unit**。

6. 点击 **Save** 完成。

过程 23.6. 用 CLI 编辑 Bean 池

1. 登录 CLI 工具并连接到您的服务器。请参考 [第 3.5.4 节 “用管理 CLI 连接受管服务器实例”](#)。
2. 对需要修改的每个属性使用 **write-attribute** 操作和下列语法。

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:write-attribute(name="ATTRIBUTE", value="VALUE")
```

- 用 Bean 池的名称替换 *BEANPOOLNAME*。
- 用要编辑的属性的名称替换 *ATTRIBUTE*。可以编辑的这些属性是 **max-pool-size**, **timeout** 和 **timeout-unit**。
- 用所需的属性值替换 *VALUE*。

3. 请用 **read-resource** 操作来确认对 Bean 池的修改。

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:read-resource
```

例 23.4. 用 CLI 设置 Bean 池的 Timeout 值

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-pool=HSBeanPool:write-attribute(name="timeout", value="1500")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[提交 bug 报告](#)

23.2.5. 为 Session 和 Message-Driven Bean 分配 Bean 池

JBoss 管理员可以通过管理控制台或 CLI 分别为 Session Bean 或 Message-driven Bean 分配独立的 Bean 池。

默认情况下有两种 Bean 池，分别用于 stateless session bean 的 **slsb-strict-max-pool** 和用于 message-driven bean 的 **mdb-strict-max-pool**。

要创建或编辑 bean 池，请参考 [第 23.2.2 节 “创建 Bean 池”](#) 和 [第 23.2.4 节 “编辑 Bean 池”](#)。

过程 23.7. 用管理控制台为 Message-Driven Bean 分配 Bean 池

1. 登陆到管理控制台。请参考 [第 3.4.2 节 “登录到管理控制台”](#)
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Container** 标签页。
3. 点击 **Edit**。
4. 从合适的组合框里为每种 Bean 选择 Bean 池。

5. 点击**Save**完成。

过程 23.8. 用 CLI 为 Message-Driven Bean 分配 Bean 池

1. 登录 CLI 工具并连接到您的服务器。请参考 [第 3.5.4 节 “用管理 CLI 连接受管服务器实例”](#)。
2. 请使用 **write-attribute** 命令和下列语法。

```
/subsystem=ejb3:write-attribute(name="BEANTYPE", value="BEANPOOL")
```

- 对于 Message-Driven Bean，用 **default-mdb-instance-pool** 替换 *BEANTYPE*，而对于 stateless session bean 则使用 **default-slsb-instance-pool**。
- 用分配的 Bean 池的名称替换 *BEANPOOL*。

3. 使用 **read-resource** 操作来确认修改。

```
/subsystem=ejb3:read-resource
```

例 23.5. 用 CLI 为 Session Bean 分配 Bean 池

```
[standalone@localhost:9999 /] /subsystem=ejb3:write-attribute(name="default-slsb-instance-pool", value="LV_SLSB_POOL")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

例 23.6. XML 配置示例

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <session-bean>
    <stateless>
      <bean-instance-pool-ref pool-name="slsb-strict-max-pool"/>
    </stateless>
    <stateful default-access-timeout="5000" cache-ref="simple"/>
    <singleton default-access-timeout="5000"/>
  </session-bean>
  <mdb>
    <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
    <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
  </mdb>
</subsystem>
```

[提交 bug 报告](#)

23.3. 配置 EJB 线程池

23.3.1. EJB 线程池

JBoss EAP 6 在内存里维护着大量的 EJB 服务使用的 Java 线程对象实例，包括远程调用、定时器服务和异步调用。

这个技术被称为线程池。它通过消除创建线程的负荷来提高性能，而且让系统管理员有了控制资源使用的机制。

用不同的参数可以创建多个线程池，且每个服务都可以分配不同的线程池。

[提交 bug 报告](#)

23.3.2. 创建线程池

EJB 线程池可以用管理控制台或 CLI 来创建。

过程 23.9. 用管理控制台来创建 EJB 线程池

1. 登陆到管理控制台。第 3.4.2 节“登录到管理控制台”
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Thread Pools** 标签页。
3. 点击 **Add** 按钮。**Add EJB3 Thread Pools** 对话框将会出现。
4. 指定所需的细节，如 **Name**、**Max. Threads** 和 **Keep-Alive Timeout** 等。
5. 点击 **Save** 完成。

过程 23.10. 用 CLI 创建线程池

1. 登录 CLI 工具并连接到您的服务器。请参考第 3.5.4 节“用管理 CLI 连接受管服务器实例”。
2. 使用 **add** 操作和下列语法。

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:add(max-threads=MAXSIZE,
keepalive-time={"time"=>"TIME", "unit"=>"UNIT"})
```

- 用线程池的名称替换 **THREADPOOLNAME**。
 - 用线程池的最大大小替换 **MAXSIZE**。
 - 用保持在线所需的时间单元替代 **UNIT**。所允许的值有：**NANOSECONDS**、**MICROSECONDS**、**MILLISECONDS**、**SECONDS**、**MINUTES**、**HOURS** 和 **DAYS**。
 - 用保持在线所需的时间替换 **TIME**。这个值是一个 **UNIT** 数字。
3. 请用 **read-resource** 操作来确认 Bean 池的创建。

```
/subsystem=ejb3/strict-max-bean-instance-pool=THREADPOOLNAME:read-resource
```

例 23.7. 用 CLI 创建线程池

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-
```

```
pool=testmepool:add(max-threads=50, keepalive-time={"time"=>"150",
"unit"=>"SECONDS"})
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

例 23.8. XML 配置示例

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">

    <thread-pools>
        <thread-pool name="default" max-threads="20" keepalive-
time="150"/>
    </thread-pools>

</subsystem>
```

[提交 bug 报告](#)

23.3.3. 删除线程池

未使用的线程池可以用管理控制台进行删除。

必须具备的条件

- 您要删除的线程池不能正在被使用。请参考下列任务以确保它没有被使用。
 - [第 23.6.2 节“配置 EJB3 定时器服务”](#)
 - [第 23.7.2 节“配置 EJB3 异步调用服务线程池”](#)
 - [第 23.8.2 节“配置 EJB3 远程服务”](#)

过程 23.11. 用管理控制台来删除 EJB 线程池

1. 登录到管理控制台。 [第 3.4.2 节“登录到管理控制台”](#)。
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Thread Pools** 标签页。
3. 选择要删除的线程池。
4. 点击 **Remove** 按钮。 **Remove Item** 对话框将会出现。
5. 点**确定**。

过程 23.12. 用 CLI 删除线程池

1. 登录 CLI 工具并连接到您的服务器。请参考 [第 3.5.4 节“用管理 CLI 连接受管服务器实例”](#)。
2. 使用 **remove** 操作和下列语法。

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:remove
```

- 用线程池的名称替换 *THREADPOOLNAME*。

例 23.9. 用 CLI 删除线程池

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-
pool=ACCTS_THREADS:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[提交 bug 报告](#)

23.3.4. 编辑线程池

JBoss 管理员可以用管理控制台或 CLI 来编辑线程池。

过程 23.13. 用管理控制台编辑线程池

1. 登录到管理控制台。第 3.4.2 节“登录到管理控制台”。
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Thread Pools** 标签页。
3. 选择要编辑的线程池。
4. 点击 **Edit**。
5. 编辑要修改的内容，您只可以修改 **Thread Factory**、**Max Threads**、**Keepalive Timeout** 和 **Keepalive Timeout Unit**。
6. 点击 **Save** 完成。

过程 23.14. 用 CLI 编辑线程池

1. 登录 CLI 工具并连接到您的服务器。请参考第 3.5.4 节“用管理 CLI 连接受管服务器实例”。
2. 对需要修改的每个线程池属性使用 **write_attribute** 操作和下列语法。

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:write-
attribute(name="ATTRIBUTE", value="VALUE")
```

- 用线程池的名称替换 *THREADPOOLNAME*。
 - 用要编辑的属性的名称替换 *ATTRIBUTE*。可以编辑的这些属性是 **keepalive-time**、**max-threads** 和 **thread-factory**。
 - 用所需的属性值替换 *VALUE*。
3. 请用 **read-resource** 操作来确认对线程池的修改。

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:read-resource
```



重要

当用 CLI 修改 **keepalive-time** 属性的值时，所需的值是一个对象形式。它具有下列语法。

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:write-attribute(name="keepalive-time", value={"time" => "VALUE", "unit" => "UNIT"})
```

例 23.10. 用 CLI 设置线程池的 Maxsize 值

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=HSThreads:write-attribute(name="max-threads", value="50") {"outcome" => "success"}
[standalone@localhost:9999 /]
```

例 23.11. 用 CLI 设置线程池的 keepalive-time 值

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=HSThreads:write-attribute(name="keepalive-time", value={"time"=>"150"}) {"outcome" => "success"}
[standalone@localhost:9999 /]
```

[提交 bug 报告](#)

23.4. 配置 SESSION BEAN

23.4.1. Session Bean 访问超时

Stateful 和 Singleton Session Bean 有一个用于管理并行访问的访问超时值。这个值指定对 Session Bean 的方法的请求超时前阻塞的时间。

超时值和单位可以用 `@javax.ejb.AccessTimeout` 注解指定。它可以在 Session Bean（应用于所有的方法）或某个方法上指定来覆盖 Bean 的配置。

如果没有指定，JBoss EAP 6 将提供一个默认的超时值 5000 毫秒。

请参考 AccessTimeout 的

Javadocs : <http://docs.oracle.com/javaee/6/api/javax/ejb/AccessTimeout.html>

[提交 bug 报告](#)

23.4.2. 设置默认的 Session Bean 访问超时时间

JBoss 管理员可以指定默认的 Singleton 和 Stateful Session Bean 的超时时间。默认值可以通过管理控制台或 CLI 设置。其默认值是 5000 毫秒。

过程 23.15. 用管理控制台设置默认的 Session Bean 访问超时时间

1. 登录到管理控制台。请参考 第 3.4.2 节 “登录到管理控制台”。
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Container** 标签页。
3. 点击 **编辑** 按钮。**Details** 区域里的字段现在可以进行编辑了。
4. 在 **Stateful Access Timeout** 和/或 **Singleton Access Timeout** 文本框里输入所需的值。
5. 点击 **Save** 完成。

过程 23.16. 用 CLI 设置 Session Bean 访问超时时间

1. 登录 CLI 工具并连接到您的服务器。请参考 第 3.5.4 节 “用管理 CLI 连接受管服务器实例”。
2. 请使用 **write-attribute** 命令和下列语法。

```
/subsystem=ejb3:write-attribute(name="BEANTYPE", value=TIME)
```

- 对于 Stateful Session Bean，用 **default-stateful-bean-access-timeout** 替换 **BEANTYPE**，而对于 Singleton Session bean 则使用 **default-singleton-bean-access-timeout**。
- 用所需的超时值替换 **TIME**。

3. 使用 **read-resource** 操作来确认修改。

```
/subsystem=ejb3:read-resource
```

例 23.12. 用 CLI 设置默认的 Session Bean 访问超时时间为 9000

```
[standalone@localhost:9999 /] /subsystem=ejb3:write-attribute(name="default-stateful-bean-access-timeout", value=9000)
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

例 23.13. XML 配置示例

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <session-bean>
    <stateless>
      <bean-instance-pool-ref pool-name="slsb-strict-max-pool"/>
    </stateless>
    <stateful default-access-timeout="5000" cache-ref="simple"/>
    <singleton default-access-timeout="5000"/>
  </session-bean>
</subsystem>
```

23.5. 配置 MESSAGE-DRIVEN BEAN

23.5.1. 为 Message-Driven Bean 设置默认的资源适配器

JBoss 管理员可以指定 Message-Driven Bean 使用的默认资源适配器。这个默认的资源适配器可以通过管理控制台或 CLI 设置。JBoss EAP 6 提供的默认值是 **hornetq-ra**。

过程 23.17. 用管理控制台为 Message-Driven Bean 设置默认的资源适配器

1. 登录到管理控制台。第 3.4.2 节“登录到管理控制台”
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Container** 标签页。
3. 点击 **编辑** 按钮。**Details** 区域里的字段现在可以进行编辑了。
4. 在 **Default Resource Adapter** 文本框里输入要使用的资源适配器名称。
5. 点击 **Save** 完成。

过程 23.18. 用 CLI 为 Message-Driven Bean 设置默认的资源适配器

1. 登录 CLI 工具并连接到您的服务器。请参考第 3.5.4 节“用管理 CLI 连接受管服务器实例”。
2. 请使用 **write-attribute** 命令和下列语法。

```
/subsystem=ejb3:write-attribute(name="default-resource-adapter-name", value="RESOURCE-ADAPTER")
```

用要使用的资源适配器名称替换 **RESOURCE-ADAPTER**。

3. 使用 **read-resource** 操作来确认修改。

```
/subsystem=ejb3:read-resource
```

例 23.14. 用 CLI 为 Message-Driven Bean 设置默认的资源适配器

```
[standalone@localhost:9999 subsystem=ejb3] /subsystem=ejb3:write-attribute(name="default-resource-adapter-name", value="EDIS-RA")
{"outcome" => "success"}
[standalone@localhost:9999 subsystem=ejb3]
```

例 23.15. XML 配置示例

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <mdb>
    <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
    <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
  </mdb>
</subsystem>
```

</subsystem>

[提交 bug 报告](#)

23.6. 配置 EJB3 定时器服务

23.6.1. EJB3 定时器服务

EJB3 定时器服务是一个标准的 Java EE 6 服务，用于调度 EJB 的方法调用。**Stateless session beans**、**singleton session beans** 和 **message-driven bean** 都可以在指定时间调度它们的回调方法。方法回调可以发生在指定的时间、以某个时间间隔循环发生或基于日历进行调度。

[提交 bug 报告](#)

23.6.2. 配置 EJB3 定时器服务

JBoss 管理员可以在管理控制台里配置 EJB3 定时器服务。可以配置的功能是用于调度的线程池和定时器所数据保存的目录。

过程 23.19. 配置 EJB3 定时器服务

1. 登录到管理控制台。请参考 [第 3.4.2 节“登录到管理控制台”](#)。
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Services** 标签页，点击 **Timer Services**。
3. 点击 **Edit** 按钮。**Details** 区域里的字段现在可以进行编辑了。
4. 如果配置了其他线程池，您可以为定时器选择不同的 EJB3 线程池；您也可以修改保存定时器服务数据的目录。定时器服务数据目录的配置由两个值组成：**Path**，数据所保存的目录；**Relative To**，包含 **Path** 的目录。在默认情况下，**Relative To** 被设置为文件系统的 **Path** 变量。
5. 点击 **Save** 完成。

[提交 bug 报告](#)

23.7. 配置 EJB 异步调用服务

23.7.1. EJB3 异步调用服务

异步调用服务（**Asynchronous Invocation Service**）是一个 EJB 容器服务，它管理 **Session Bean** 方法的异步调用。这个服务维护着可配置的数量线程（线程池），分配给异步方法调用。

Enterprise JavaBeans 3.1 允许对 **Session Bean**（**stateful**、**stateless** 或 **singleton**）的任何方法进行注解以允许异步执行。

[提交 bug 报告](#)

23.7.2. 配置 EJB3 异步调用服务线程池

JBoss 管理员可以在 JBoss EAP 6 管理控制台里配置 EJB3 异步调用服务来使用专用的线程池。

过程 23.20. 配置 EJB3 异步调用服务线程池

1. 登录到管理控制台。请参考 [第 3.4.2 节“登录到管理控制台”](#)。
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Services** 标签页，点击 **Async Service**。
3. 点击 **Edit**。
4. 从列表里选择要使用的 EJB3 线程池。这个线程池必须是已创建的线程池。
5. 点击 **Save** 完成。

[提交 bug 报告](#)

23.8. 配置 EJB3 远程调用服务

23.8.1. EJB3 远程服务

EJB3 远程服务管理带有远程商业接口的 EJB 的远程执行。

[提交 bug 报告](#)

23.8.2. 配置 EJB3 远程服务

JBoss 管理员可以在管理控制台里配置 EJB3 Remote 服务。可以配置的功能是用于远程调用的线程池和 EJB3 Remoting 频道所注册的连接器。

过程 23.21. 配置 EJB3 远程服务

1. 登录到管理控制台。请参考 [第 3.4.2 节“登录到管理控制台”](#)。
2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **EJB 3**。然后选择 **Services** 标签页，点击 **Remote Service**。
3. 点击 **Edit**。
4. 如果配置了其他线程池，您可以为远程服务选择不同的 EJB3 线程池；您也可以修改用来注册 EJB Remoting 频道的连接器。
5. 点击 **Save** 完成。

[提交 bug 报告](#)

23.9. 配置 EJB 2.X ENTITY BEANS

23.9.1. EJB Entity Bean

EJB Entity Bean 是 EJB 2.x 规格的类型，它表示保存在数据库里的持久性数据。Entity Bean 已被 JPA Entity 代替，且在 EJB 官方规格里已列为删除（削减）内容。Red Hat 不推荐使用 Entity Bean，除非是为了向后兼容。

在默认情况下，JBoss EAP 6 是禁用对 Entity Bean 的支持的。

[提交 bug 报告](#)

23.9.2. Container-Managed Persistence（容器管理的持久化）

容器管理的持久化（Container-Managed Persistence，CMP）是应用服务器提供的一个服务，它为 Entity Bean 提供数据持久化。

[提交 bug 报告](#)

23.9.3. 启用 EJB 2.x 的容器管理持久化

容器管理持久化（Container-Managed Persistence，CMP）是由 `org.jboss.as.cmp` 扩展处理的。受管域和独立服务器的 full 配置（`standalone-full.xml`）里 CMP 默认是启用的。

要在不同的配置里启用 CMP，请在服务器配置文件的启用扩展列表里添加 `org.jboss.as.cmp` 模块。

```
<extensions>
    <extension module="org.jboss.as.cmp"/>
</extensions>
```

添加扩展后，您必须在配置集的 XML 文件里的 `<profile>` 元素下添加下列元素。

```
<subsystem xmlns="urn:jboss:domain:cmp:1.1"/>
```

要在服务器配置里禁用 CMP，请从 `org.jboss.as.cmp` 模块里删除扩展条目。

[提交 bug 报告](#)

23.9.4. 配置 EJB 2.x 的容器管理持久化

您可以配置 EJB 2.x 的容器管理持久化子系统以指定任何数量的密钥生成器。密钥生成器用于生成唯一的密钥以标识 CMP 服务持久化的每个实体。

您可以定义两种类型的密钥生成器：UUID 和 HiLo 密钥生成器。

基于 UUID 的生成器

基于 UUID 的生成器可用 UUID 创建密钥。基于 UUID 的生成器只需要具有唯一的名称，它不需要其他配置。

基于 UUID 的密钥生成器可以使用下列语法通过 CLI 添加：

```
/subsystem=cmp/uuid-keygenerator=UNIQUE_NAME:add
```

例 23.16. 添加 UUID 密钥生成器

要添加名为 `uuid_identities` 的基于 UUID 的密钥生成器，请使用这个 CLI 命令：

```
/subsystem=cmp/uuid-keygenerator=uuid_identities:add
```

这个命令创建的 XML 配置是：

```
<subsystem xmlns="urn:jboss:domain:cmp:1.0">
    <key-generators>
```

```

        <uuid name="uuid_identities" />
      </key-generators>
    </subsystem>

```

HiLo 密钥生成器

HiLo 密钥生成器使用数据库来创建和存储标识密钥。HiLo 密钥生成器必须具有唯一的名称并配置指定用来存储数据、表名及存储密钥的字段的数据源的属性。

HiLo 密钥生成器可以使用下列语法通过 CLI 添加：

```

/subsystem=cmp/hilo-keygenerator=UNIQUE_NAME/:add(property=value,
property=value, ...)

```

例 23.17. 添加 HiLo 密钥生成器

```

/subsystem=cmp/hilo-keygenerator=HiLoKeyGeneratorFactory:add(create-
table=true,create-table-ddl="create table HILOSEQUENCES (SEQUENCENAME
varchar(50) not null, HIGHVALUES integer not null, constraint hilo_pk
primary key (SEQUENCENAME))",data-
source=java:jboss/datasources/ExampleDS, id-
column=HIGHVALUES,sequence-column=SEQUENCENAME,table-
name=HILOSEQUENCES,sequence-name=general,block-size=10)

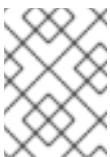
```

这个命令创建的 XML 配置是：

```

<subsystem xmlns="urn:jboss:domain:cmp:1.1">
  <key-generators>
    <hilo name="HiLoKeyGeneratorFactory">
      <block-size>10</block-size>
      <create-table>true</create-table>
      <create-table-ddl>create table HILOSEQUENCES
(SEQUENCENAME varchar(50) not null, HIGHVALUES integer not null,
constraint hilo_pk primary key (SEQUENCENAME))</create-table-ddl>
      <data-source>java:jboss/datasources/ExampleDS</data-
source>
      <id-column>HIGHVALUES</id-column>
      <sequence-column>SEQUENCENAME</sequence-column>
      <sequence-name>general</sequence-name>
      <table-name>HILOSEQUENCES</table-name>
    </hilo>
  </key-generators>
</subsystem>

```



注意

block-size 必须设置为 **value != 0**，否则生成的 PKey 不会递增，因此实体的创建会失败且抛出 **DuplicateKeyException**。



注意

使用群集时 `select-hi-ddl` 必须设置为 'FOR UPDATE' 以确保一致性。所有的数据库都不支持锁定功能。

[提交 bug 报告](#)

23.9.5. HiLo 密钥生成器使用的 CMP 子系统属性

表 23.1. HiLo 密钥生成器使用的 CMP 子系统属性

属性	数据类型	描述
<code>block-size</code>	long	块大小。
<code>create-table</code>	boolean	如果为 <code>TRUE</code> 且名为 <code>table-name</code> 的表不存在，这个表将用 <code>create-table-ddl</code> 的内容创建。
<code>create-table-ddl</code>	string	用来创建 <code>table-name</code> 里指定的表的 DDL 命令（如果这个表不存在且 <code>create-table</code> 为 <code>TRUE</code> ）。
<code>data-source</code>	token	用于连接数据库的数据源。
<code>drop-table</code>	boolean	指定是否删除表。
<code>id-column</code>	token	ID 列的名称。
<code>select-hi-ddl</code>	string	将返回当前存储的最大密钥的 SQL 命令。
<code>sequence-column</code>	token	Sequence 列的名称。
<code>sequence-name</code>	token	序列的名称。
<code>table-name</code>	token	用于存储密钥信息的表的名称。

[提交 bug 报告](#)

第 24 章 JAVA 连接器架构（JCA）

24.1. 介绍

24.1.1. 关于 Java EE Connector API (JCA)

JBoss EAP 6 提供了对 Java EE Connector API (JCA) 1.6 规格的完整支持。关于 JCA 规格的更多信息，请参考 [JSR 322: Java EE Connector Architecture 1.6](#)。

资源适配器（Resource Adapter）是一个实现 Java EE Connector API (JCA) 架构的组件。它和数据源对象类似，但它提供了从企业新系统（EIS）和多种系统间的连接性，如数据库、消息系统、事务处理和企业资源计划系统（ERP）。

[提交 bug 报告](#)

24.1.2. JAVA 连接器架构（JCA）

Java EE 连接器架构（JCA）定义了 Java EE 系统到外部各种企业信息系统（EIS）的标准架构。EIS 的例子包括 EARP 系统、大型机事务处理（TP）、数据库和消息系统。

JCA 1.6 提供了下列管理功能：

- 连接
- 事务
- 安全性
- 生命周期
- 工作实例
- 事务流入
- 消息流入

JCA 1.6 是基于 JSR-322 <http://jcp.org/en/jsr/detail?id=313> 开发的。

[提交 bug 报告](#)

24.1.3. 资源适配器

资源适配器是一个可部署的 Java EE 组件，它通过 Java 连接器架构（JCA）提供 Java EE 应用程序和企业信息系统（EIS）之间的通讯。资源适配器通常由 EIS 供应商提供以便于它们的产品和 Java EE 应用程序的集成。

企业信息系统（EIS）可以是机构内部的任何软件系统。如 EAR、数据库系统、电子邮件服务器和私有消息系统。

资源适配器打包在可以部署到 JBoss EAP 6 的 RAR 文件里。RAR 文件也可以包含在 EAR 部署文件里。

[提交 bug 报告](#)

24.2. 配置 JAVA 连接器架构 (JAVA CONNECTOR ARCHITECTURE, JCA) 子系统

JBoss EAP 6 配置文件里的 JCA 子系统控制 JCA 容器和资源适配器部署的普通设置。

JCA 子系统的关键元素

归档检验

- 它设置了在部署单元上执行的归档检验。
- 下表描述了您可以为归档检验设置的属性。

表 24.1. 归档检验属性

属性	默认值	描述
enabled	true	指定是否启用归档检验
fail-on-error	true	指定部署失败时是否报告归档检验错误。
fail-on-warn	false	指定部署失败时是否报告归档检验警告。

- 如果归档没有正确实现 Java EE 连接器架构规格且启用了归档检验，在部署期间会出现描述这个问题的错误信息。例如：

```
Severity: ERROR
Section: 19.4.2
Description: A ResourceAdapter must implement a "public int hashCode()" method.
Code: com.mycompany.myproject.ResourceAdapterImpl

Severity: ERROR
Section: 19.4.2
Description: A ResourceAdapter must implement a "public boolean equals(Object)" method.
Code: com.mycompany.myproject.ResourceAdapterImpl
```

- 如果没有指定归档检验，它会被认作已存在且 **enabled** 默认为 True。

Bean 检验

- 这个设置确定 Bean 检验 (JSR-303) 是否在部署单元上执行。
- 下表描述了您可以为 Bean 检验设置的属性。

表 24.2. Bean 检验属性

属性	默认值	描述
enabled	true	指定是否启用 Bean 检验。

- 如果没有指定 Bean 检验，它会被认作已存在且 **enabled** 默认为 True。

Work Manager

- 有两种 Work Manager：

默认的 Work Manager

默认的 Work Manager 及其线程池。

自定义 Work Manager

自定义 Work Manager 定义及其线程池。

- 下表描述了您可以为 Work Manager 设置的属性。

表 24.3. Work Manager 的属性

属性	描述
name	指定 Work Manager 的名称。对于自定义 Work Manager 这是必需的。
short-running-threads	标准 Work 实例的线程池。每个 Work Manager 都有一个短期运行的线程池。
long-running-threads	用于设置 LONG_RUNNING 的 JCA 1.6 Work 实例的线程池。每个 Work Manager 都有一个可选的长期期运行的线程池。

- 下表描述了您可以为 Work Manager 线程池设置的属性。

表 24.4. 线程池的属性

属性	描述
allow-core-timeout	Boolean 值，确定核心线程是否超时。默认为 False。
core-threads	核心线程池的大小。它必须比线程池的最大尺寸要小。
queue-length	队列的最大长度。
max-thread	线程池的最大尺寸。
keepalive-time	指定在开始工作后池线程应该保持的时间。
thread-factory	对线程工厂的引用。

Bootstrap contexts

- 用来定义自定义引导上下文。
- 下表描述了您可以为引导上下文设置的属性。

表 24.5. 引导上下文属性

属性	描述
name	指定引导上下文的名称。
workmanager	指定用于这个上下文的 Work Manager 的名称。

缓存的连接管理者

- 用于调试连接和支持事务里的 Lazy 连接建立，跟踪应用程序是否使用或正确释放它们。
- 下表描述您可以为缓存连接管理者设置的属性。

表 24.6. 缓存连接管理者的属性

属性	默认值	描述
debug	false	显性关闭连接失败时输出警告
error	false	显性关闭连接失败时抛出异常

过程 24.1. 用管理控制台配置 JCA 子系统

JBoss EAP 6 的 JCA 子系统可以在管理控制台里进行配置。根据服务器运行的方式，JCA 配置选项在管理控制台里所处的位置会有轻微的不同。

1. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connector** 菜单并选择 **JCA**。
2. 如果服务器运行于域模式，请从左上角的 **Profile** 下拉菜单里选择合适的配置集。
3. 用下列三个标签页配置 JCA 子系统。
 - a. **Common Config**
Common Config 标签页包含每个缓存连接管理者、归档检验和 Bean 检验 (JSR-303) 的设置。这些设置都包含在自己的标签页里。您可以打开合适的标签页并点击 **Edit** 按钮，进行所需修改后再 **Save** 按钮保存。

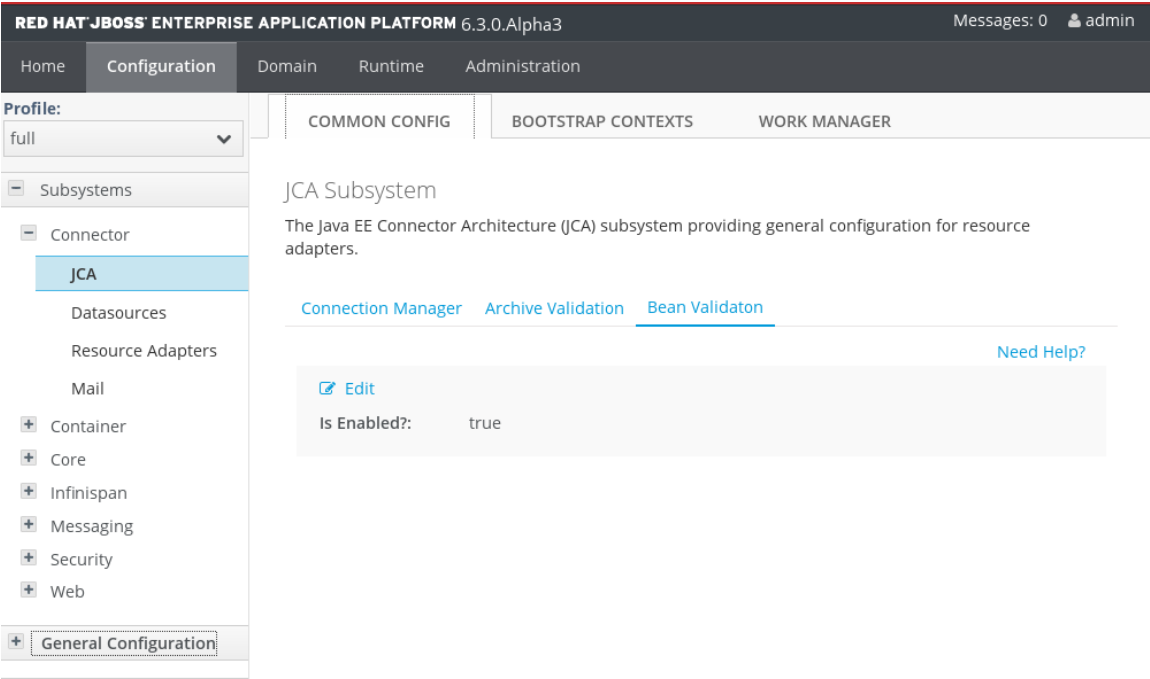


图 24.1. JCA 的常见配置

b. Work Managers

Work Manager 标签页包含配置的 Work Manager 的列表。您可以新建、删除 Work Manager 以及配置线程池。每个 Work Manager 都可以有一个短期运行以及一个长期运行的线程池。

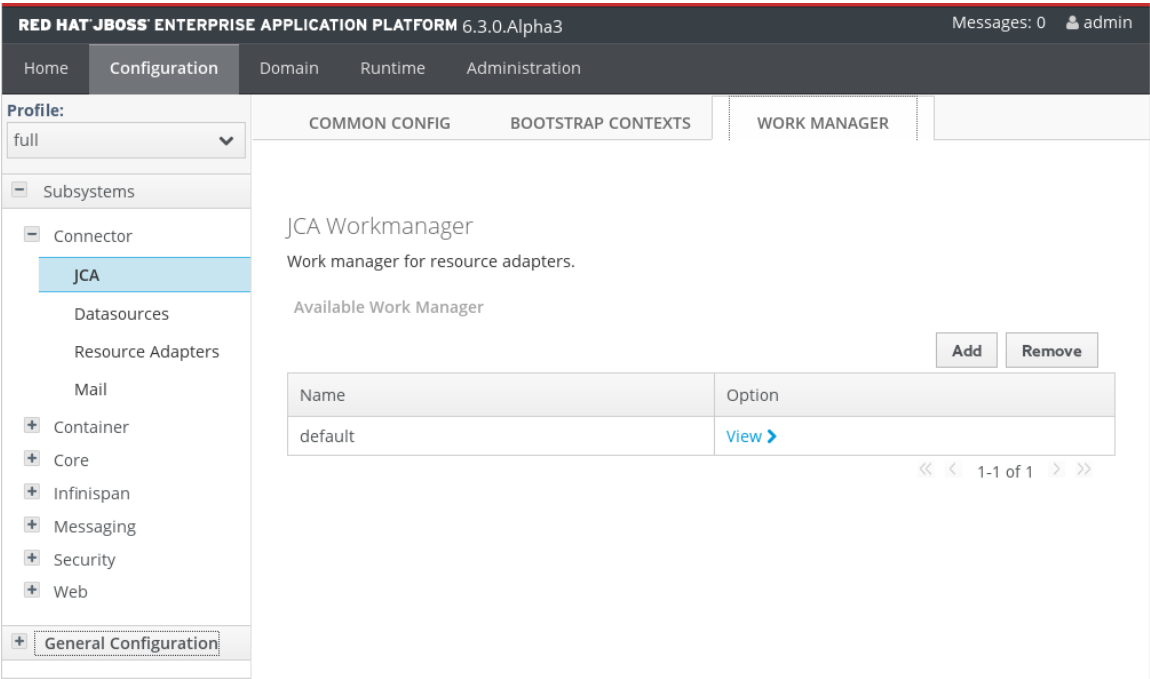


图 24.2. Work Managers

线程池的属性可以通过点击所选资源适配器的 **View** 来配置。

The screenshot shows the 'Thread Pools' configuration page in the JBoss configuration console. The left sidebar shows the 'Subsystems' tree with 'JCA' selected. The main content area has tabs for 'COMMON CONFIG', 'BOOTSTRAP CONTEXTS', and 'WORK MANAGER'. The 'Thread Pools' page displays the 'Work Manager: default' and a table of 'Available Thread Pools'.

Name	Type	Max Threads
default	short-running	50
default	long-running	50

Below the table, there are links for 'Attributes' and 'Sizing'. An 'Edit' button is visible for the 'default' thread pool configuration.

图 24.3. Work Manager 的线程池

c. Bootstrap Contexts

Bootstrap Contexts 标签页包含了 **Bootstrap Contexts** 列表。您可以创建、删除和配置 Bootstrap Context。每个 Bootstrap Context 都必须分配一个 Work Manager。

The screenshot shows the 'Bootstrap Contexts' configuration page in the JBoss configuration console. The left sidebar shows the 'Subsystems' tree with 'JCA' selected. The main content area has tabs for 'COMMON CONFIG', 'BOOTSTRAP CONTEXTS', and 'WORK MANAGER'. The 'Bootstrap Contexts' page displays the 'JCA Bootstrap Contexts' and a table of 'Available Bootstrap Context'.

Name
default

Below the table, there are links for 'Selection' and 'Need Help?'. An 'Edit' button is visible for the 'default' bootstrap context configuration.

图 24.4. Bootstrap Contexts

[提交 bug 报告](#)

24.3. 部署资源适配器

资源适配器可以用管理 CLI 工具、基于 Web 的管理控制台或手动复制文件的方式部署到 JBoss EAP 6。这个过程和其他可部署的 Artifact 是一样的。

过程 24.2. 用管理 CLI 部署资源适配器

1. 在你的操作系统里打开一个命令行提示窗口。
2. 连接至管理 CLI。

- 对于 Linux，输入下列命令：

```
$ EAP_HOME/bin/jboss-cli.sh --connect  
$ Connected to standalone controller at localhost:9999
```

- 对于 Windows，输入下列命令：

```
C:\>EAP_HOME\bin\jboss-cli.bat --connect  
C:\> Connected to standalone controller at localhost:9999
```

3. 部署资源适配器。

- 要部署资源适配器到独立服务器，请输入下列命令：

```
$ deploy path/to/resource-adapter-name.rar
```

- 要部署资源适配器到受管域里的所有服务器组，请输入下列命令：

```
$ deploy path/to/resource-adapter-name.rar --all-server-groups
```

过程 24.3. 用管理控制台部署资源适配器

1. 登录到管理控制台。请参考 [第 3.4.2 节“登录到管理控制台”](#)。
2. 点击屏幕顶部的 **Runtime** 标签页并选择 **Manage Deployments**。然后点击 **Add**。
3. 浏览资源适配器并选择它。然后点击 **Next**。
4. 验证部署名称，然后点击 **Save**。
5. 现在资源适配器归档应该以已禁用的状态显示在列表里。
6. 启用资源适配器。
 - 在域模式里，点击 **Assign**。选择将资源适配器分配给哪个服务器组。点击 **Save** 完成。
 - 在独立模式里，从列表里选择 **Application Component**。点击 **En/Disable**。然后点击 **Are You Sure?** 对话框里的 **Confirm** 来启用组件。

过程 24.4. 手动部署资源适配器

- 复制资源适配器归档到服务器的部署目录，

- 对于独立服务器，复制资源适配器归档到 ***EAP_HOME/standalone/deployments/*** 目录。
- 对于受管域，您必须使用管理控制台或管理 CLI 将资源适配器归档部署到服务器组。

[提交 bug 报告](#)

24.4. 配置已部署的资源适配器

JBoss 管理员可以用管理 CLI 工具、基于 Web 的管理控制台或手动编辑配置文件的方式来配置资源适配器。

关于所支持的属性和其他细节，请参考资源适配器的相关文档。



注意

在下面的过程里，请在 [**standalone@localhost:9999 /**] 提示符后输入命令行。请不要输入花括号里的内容。这是命令执行的结果，例如 {"outcome" => "success"}。

过程 24.5. 用管理 CLI 配置资源适配器

1. 在你的操作系统里打开一个命令行提示窗口。
2. 连接至管理 CLI。

- 对于 Linux，输入下列命令：

```
$ EAP_HOME/bin/jboss-cli.sh --connect
```

您应该看到下列输出结果：

```
$ Connected to standalone controller at localhost:9999
```

- 对于 Windows，输入下列命令：

```
C:\>EAP_HOME\bin\jboss-cli.bat --connect
```

您应该看到下列输出结果：

```
C:\> Connected to standalone controller at localhost:9999
```

3. 添加资源适配器配置。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar:add(archive=eis.rar, transaction-support=XATransaction)
{"outcome" => "success"}
```

4. 配置 **server** 资源适配器级别 <config-property>。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar/config-properties=server/:add(value=localhost)
{"outcome" => "success"}
```

5. 配置 **port** 资源适配器级别 **<config-property>**。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/config-properties=port/:add(value=9000)
{"outcome" => "success"}
```

6. 为受管连接工厂添加连接定义。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/connection-definitions=cfName:add(class-
name=com.acme.eis.ra.EISManagedConnectionFactory, jndi-
name=java:/eis/AcmeConnectionFactory)
{"outcome" => "success"}
```

7. 配置 **name** 受管连接工厂级别的 **<config-property>**。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/connection-definitions=cfName/config-
properties=name/:add(value=Acme Inc)
{"outcome" => "success"}
```

8. 添加 **admin** 对象。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/admin-objects=aoName:add(class-
name=com.acme.eis.ra.EISAdminObjectImpl, jndi-
name=java:/eis/AcmeAdminObject)
{"outcome" => "success"}
```

9. 配置 **Admin** 对象的 **threshold** 属性。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/admin-objects=aoName/config-
properties=threshold/:add(value=10)
{"outcome" => "success"}
```

10. 激活资源适配器。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar:activate
{"outcome" => "success"}
```

11. 查看最新配置和激活的资源适配器。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar:read-resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "archive" => "eis.rar",
    "beanvalidationgroups" => undefined,
    "bootstrap-context" => undefined,
```



```

"transaction-support" => "XATransaction",
"admin-objects" => {"aoName" => {
    "class-name" => "com.acme.eis.ra.EISAdminObjectImpl",
    "enabled" => true,
    "jndi-name" => "java:/eis/AcmeAdminObject",
    "use-java-context" => true,
    "config-properties" => {"threshold" => {"value" => 10}}
}},
"config-properties" => {
    "server" => {"value" => "localhost"},
    "port" => {"value" => 9000}
},
"connection-definitions" => {"cfName" => {
    "allocation-retry" => undefined,
    "allocation-retry-wait-millis" => undefined,
    "background-validation" => false,
    "background-validation-millis" => undefined,
    "blocking-timeout-wait-millis" => undefined,
    "class-name" =>
"com.acme.eis.ra.EISManagedConnectionFactory",
    "enabled" => true,
    "flush-strategy" => "FailingConnectionOnly",
    "idle-timeout-minutes" => undefined,
    "interleaving" => false,
    "jndi-name" => "java:/eis/AcmeConnectionFactory",
    "max-pool-size" => 20,
    "min-pool-size" => 0,
    "no-recovery" => undefined,
    "no-tx-separate-pool" => false,
    "pad-xid" => false,
    "pool-prefill" => false,
    "pool-use-strict-min" => false,
    "recovery-password" => undefined,
    "recovery-plugin-class-name" => undefined,
    "recovery-plugin-properties" => undefined,
    "recovery-security-domain" => undefined,
    "recovery-username" => undefined,
    "same-rm-override" => undefined,
    "security-application" => undefined,
    "security-domain" => undefined,
    "security-domain-and-application" => undefined,
    "use-ccm" => true,
    "use-fast-fail" => false,
    "use-java-context" => true,
    "use-try-lock" => undefined,
    "wrap-xa-resource" => true,
    "xa-resource-timeout" => undefined,
    "config-properties" => {"name" => {"value" => "Acme
Inc"}}
    }}
    }
}

```

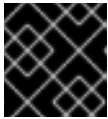
过程 24.6. 用基于 Web 的管理控制台配置资源适配器

1. 登录到管理控制台。请参考 第 3.4.2 节 “登录到管理控制台”。

2. 点击屏幕顶部的 **Configuration** 标签页。展开 **Connectors** 菜单并选择 **Resource Adapters**。
 - a. 对于域模式，从左上角的下拉菜单里选择 **Profile**。
- 点添加。
3. 输入归档名称并从 **TX:** 下拉框里选择事务类型 **XATransaction**。然后点击 **Save**。
4. 选择 **Properties** 标签页，然后点击 **Add**。
5. 在 **Name** 和主机名里输入 **server**，例如在 **Save** 里输入 **localhost**。然后点击 **Save** 完成。
6. 再次点击 **Add**。输入 **Name** 和 **port** 号码，例如在 **Value** 里输入 **9000**。然后点击 **Save** 完成。
7. **server** 和 **port** 属性现在出现在 **Properties** 面板里。点击列出的资源适配器的 **Option** 列下的 **View** 链接来查看 **Connection Definitions**。
8. 点击 **Available Connection Definitions** 表上的 **Add** 添加连接定义。
9. 输入 **Connection Class** 的 **JNDI Name** 和全限定类名。然后点击 **Save** 完成。
10. 选择新的连接定义，然后选择 **Properties** 标签页。点击 **Add** 后输入这个连接定义的 **Key** 和 **Value** 数据。点击 **Save** 完成。
11. 连接定义已完成，但却是禁用的。选择连接定义并点击 **Enable** 启用它。
12. **Really modify Connection Definition?** 对话框会出现询问是否进行修改。请点击 **Confirm**。连接定义现在应该显示为 **Enabled**。
13. 点击页面顶部的 **Admin Objects** 标签来创建和配置 **admin** 对象。然后点击 **Add** 按钮。
14. 输入 **admin** 对象的 **JNDI Name** 和全限定的 **Class Name**。然后点击 **Save**。
15. 选择 **Properties** 标签页，然后点击 **Add** 来添加 **admin** 对象的属性。
16. 在 **Name** 字段里输入一个 **admin** 对象配置属性，例如 **threshold**。在 **Value** 字段里输入属性值，如 **10**。然后点击 **Save** 保存属性。
17. **admin** 对象已完成，但却是禁用的。请点击 **Enable** 启用它。
18. **Really modify Admin Object?** 对话框会出现询问是否进行修改。请点击 **Confirm**。**admin** 对象现在应该显示为 **Enabled**。
19. 您必需重载服务器配置来完成这个过程。点击 **Runtime** 视图。展开 **Server** 菜单。选择左侧导航面板上的 **Overview**。
 - a. 重启服务器
 - 在域模式里，在服务器组上悬停鼠标。选择 **Restart Group**。
 - 在独立模式里，您可以使用 **Reload** 按钮。点击 **Reload**。
20. **Do you want to reload the server configuration?** 对话框会出现询问是否进行重载。请点击 **Confirm**。服务器配置现在是最新版本了。

过程 24.7. 手动配置资源适配器

1. 停止 JBoss EAP 6 服务器。

**重要**

要使修改在服务器重启后仍然生效，你必须在编辑服务器配置文件前停止服务器。

2. 打开服务器配置文件进行编辑。
 - 对于独立服务器，这个文件是 **`EAP_HOME/standalone/configuration/standalone.xml`**。
 - 对于受管域，这个文件是 **`EAP_HOME/domain/configuration/domain.xml`**。
3. 在配置文件里找到 **`urn:jboss:domain:resource-adapters`** 子系统。
4. 如果没有为这个子系统定义资源适配器，请首先替换：

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1"/>
```

使用下列内容：

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">
  <resource-adapters>
    <!-- <resource-adapter> configuration listed below -->
  </resource-adapters>
</subsystem>
```

5. 用您的资源适配器的 XML 定义替换 **`<!-- <resource-adapter> configuration listed below -->`**。下面是用管理 CLI 和基于 Web 的管理控制台创建的资源适配器配置的 XML 片段。

```
<resource-adapter>
  <archive>
    eis.rar
  </archive>
  <transaction-support>XATransaction</transaction-support>
  <config-property name="server">
    localhost
  </config-property>
  <config-property name="port">
    9000
  </config-property>
  <connection-definitions>
    <connection-definition class-
name="com.acme.eis.ra.EISManagedConnectionFactory"
    jndi-name="java:/eis/AcmeConnectionFactory"
```

```
        pool-name="java:/eis/AcmeConnectionFactory">
    <config-property name="name">
        Acme Inc
    </config-property>
</connection-definition>
</connection-definitions>
<admin-objects>
    <admin-object class-
name="com.acme.eis.ra.EISAdminObjectImpl"
        jndi-name="java:/eis/AcmeAdminObject"
        pool-name="java:/eis/AcmeAdminObject">
        <config-property name="threshold">
            10
        </config-property>
    </admin-object>
</admin-objects>
</resource-adapter>
```

6. 期待服务器
重新启动 JBoss EAP 6 服务器以使用新的配置运行。

[提交 bug 报告](#)

24.5. 资源适配器描述符参考

下表描述了资源适配器描述符文件的元素。

表 24.7. 主要元素

元素	描述
bean-validation-groups	指定应该使用的 Bean 检验组
bootstrap-context	指定应该使用的引导上下文的唯一名称。
config-property	指定资源适配器的配置属性。
transaction-support	定义这个资源适配器支持的事务类型。有效的值为： NoTransaction 、 LocalTransaction 、 XATransaction 。
connection-definitions	指定连接定义
admin-objects	指定管理对象

表 24.8. Bean 检验组元素

元素	描述
bean-validation-group	指定用于检验的 Bean 检验组的全限定类名

表 24.9. 连接定义 / Admin 对象的属性

属性	描述
class-name	指定受管连接工厂或 Admin 对象的全限定类名。
jndi-name	指定 JNDI 名称
enabled	对象是否被激活
use-java-context	指定是否使用 java: / JNDI 上下文
pool-name	指定对象的池名
use-ccm	启用缓存的连接管理者

表 24.10. 连接定义元素

元素	描述
config-property	指定受管连接工厂的配置属性。
pool	指定池的设置
xa-pool	指定 XA 池的设置
security	指定安全设置
timeout	指定超时设置
validation	指定检验设置
recovery	指定 XA 恢复设置

表 24.11. 池元素

元素	描述
min-pool-size	指定池应该保持的连接的最小数目。在连接的请求主题知晓之前，这些连接不会被创建。默认值是 0。

元素	描述
max-pool-size	指定某个池的连接的最大数目。每个子池里不会创建超过 max-pool-size 个连接。默认值是 20 。
prefill	是否预填充连接池。默认值是 false 。
use-strict-min	指定是否严格考虑 min-pool-size。默认值为 false 。
flush-strategy	指定在发生错误时如何冲刷池。有效值为： FailingConnectionOnly (默认值)、 IdleConnections 、 EntirePool 。

表 24.12. XA 池元素

元素	描述
min-pool-size	指定池应该保持的连接的最小数目。在连接的请求主题知晓之前，这些连接不会被创建。默认值是 0 。
max-pool-size	指定某个池的连接的最大数目。每个子池里不会创建超过 max-pool-size 个连接。默认值是 20 。
prefill	是否预填充连接池。默认值是 false 。
use-strict-min	指定是否严格考虑 min-pool-size。默认值为 false 。
flush-strategy	指定在发生错误时如何冲刷池。有效值为： FailingConnectionOnly (默认值)、 IdleConnections 、 EntirePool 。
is-same-rm-override	is-same-rm-override 元素允许您无条件地设置 <code>javax.transaction.xa.XAResource.isSameRM(XAResource)</code> 返回 true 或 false 。
interleaving	为 XA 连接工厂启用 interleaving 的元素
no-tx-separate-pools	Oracle 不允许既在 JTA 事务内部又在外部使用 XA 连接。变通办法是为不同的上下文创建单独的子池。
pad-xid	是否保护 (Pad) XID
wrap-xa-resource	是否将 XAResource 实例包裹在 <code>org.jboss.tm.XAResourceWrapper</code> 实例里

表 24.13. 安全元素

元素	描述
application	指定应用程序提供的参数（如来自 <code>getConnection(user, pw)</code> ）是否可用于区分池里的连接。
security-domain	指定用于区分池里连接的主题（来自安全域）。 <code>security-domain</code> 的内容是处理验证的 JAAS 安全管理者的名称。这个名称和 <code>JAAS login-config.xml</code> 的 <code>application-policy/name</code> 属性是关联的。
security-domain-and-application	指定用于区分池里连接的应用程序参数（如来自 <code>getConnection(user, pw)</code> ）或主题（来自安全域）。 <code>security-domain</code> 的内容是处理验证的 JAAS 安全管理者的名称。这个名称和 <code>JAAS login-config.xml</code> 的 <code>application-policy/name</code> 属性是关联的。

表 24.14. 超时元素

元素	描述
blocking-timeout-millis	指定在抛出异常前等待连接时阻塞的最长时间。请注意，这只是等待连接许可时的阻塞，如果创建连接时超时并不会抛出异常。默认值是 30000 （30 秒）。
idle-timeout-minutes	指定连接在被关闭前最长的空闲时间。实际的最长时间也取决于 <code>IdleRemover</code> 扫描时间，它是所有池里最小的 <code>idle-timeout-minutes</code> 的二分之一。
allocation-retry	指定在抛出异常前尝试分配连接的次数。默认值为 0 。
allocation-retry-wait-millis	指定试图分配连接时重试的间隔（毫秒）。默认值是 5000 （5 秒）。
xa-resource-timeout	传递给 <code>XAResource.setTransactionTimeout()</code> 的参数。默认为 0 ，表示不调用 <code>setter</code> 。其单位为秒。

表 24.15. 检验元素

元素	描述
background-validation	指定这个连接在使用前应该通过后台线程进行检验。
background-validation-minutes	<code>background-validation-minutes</code> 元素
use-fast-fail	在第一次尝试时如果连接无效则失败（ <code>true</code> ）、或者一直尝试直至遍历连接池里所有连接（ <code>false</code> ）。默认为 false 。

表 24.16. Admin 对象的元素

元素	描述
<code>config-property</code>	指定管理对象的配置属性。

表 24.17. 恢复元素

元素	描述
<code>recover-credential</code>	指定应用于恢复的用户名/密码或安全域。
<code>recover-plugin</code>	指定 <code>org.jboss.jca.core.spi.recovery.RecoveryPlugin</code> 类的实现。

定义在 `jboss-as-resource-adapters_1_0.xsd` 里的部署模式以及用于自动激活的 http://www.ironjacamar.org/doc/schema/ironjacamar_1_0.xsd。

[提交 bug 报告](#)

24.6. 查看定义的连接统计

您可以在 `deployment=name.rar` 子树里查看定义连接的统计信息。

统计信息在这个级别而不是 `/subsystem` 级别来定义是为了确保可以被没有在 `standalone.xml` 或 `domain.xml` 文件里的任何配置里定义的 `rar` 访问。

例如：

例 24.1.

```
/deployment=example.rar/subsystem=resource-adapters/statistics=statistics/connection-definitions=java\:\testMe:read-resource(include-runtime=true)
```



注意

确保您指定了 `include-runtime=true` 参数，因为所有统计都是 `runtime` 信息。默认值为 `false`。

[提交 bug 报告](#)

24.7. 资源适配器的统计信息

核心统计信息

下表包含了所支持的资源适配器核心统计信息：

表 24.18. 核心统计信息

实例类型	描述
ActiveCount	活动连接的数量。每个连接都是正在被应用程序使用或是在池里备用。
AvailableCount	池里可用连接的数量。
AverageBlockingTime	获取池里排他锁时阻塞的平均时间。单位为毫秒。
AverageCreationTime	创建连接所花费的平均时间。单位为毫秒。
CreatedCount	创建的连接数量。
DestroyedCount	销毁的连接数量。
InUseCount	正在使用的连接的数量。
MaxCreationTime	创建连接所花费的最长时间。单位为毫秒。
MaxUsedCount	使用的连接的最大数目。
MaxWaitCount	同一时间等待连接的请求的最大数目。
MaxWaitTime	等待池里排他锁所花费的最长时间。
TimedOut	超时连接的数量。
TotalBlockingTime	等待池里排他锁总共所花费的时间。单位为毫秒。
TotalCreationTime	创建连接总共所花费的时间。单位为毫秒。
WaitCount	需要等待连接的请求的数量。

[提交 bug 报告](#)

24.8. 部署 WEBSPHERE MQ 资源适配器

关于 WebSphere MQ

WebSphere MQ 是 IBM 的面向消息中间件 (Messaging Oriented Middleware, MOM)，它允许分布式系统上的应用程序彼此通讯。这是通过消息和消息队列来实现的。WebSphere MQ 负责递送消息到消息队列并用消息频道传输数据到其他队列管理者。关于 WebSphere MQ 的更多信息，请参考 [WebSphere MQ](#)。

介绍

本节涵盖在 JBoss EAP 6 里部署和配置 WebSphere MQ 资源适配器的步骤。您可以手动编辑配置文件来完成，也可以通过管理 CLI 或使用基于 Web 的管理控制台来完成。

前提条件

在开始之前，您必需检验 WebSphere MQ 资源适配器的版本并理解一些 WebSphere MQ 配置属性。

- WebSphere MQ 资源适配器是作为 Resource Archive (RAR) 文件 (`wmq.jmsra-VERSION.rar`) 提供的。您必须使用 **7.5.0.0** 或之后的版本。
- 您必须了解下列 WebSphere MQ 配置属性。关于这些属性的细节，请参考 WebSphere MQ 产品文档。
 - `MQ.QUEUE.MANAGER` : WebSphere MQ 队列管理者的名称
 - `MQ.PORT` : 用来连接 WebSphere MQ 队列管理者的主机名
 - `MQ.PORT` : 用来连接 WebSphere MQ 队列管理者的服务器频道
 - `MQ.QUEUE.NAME` : 目标队列的名称
 - `MQ.TOPIC.NAME` : 目标主题的名称
 - `MQ.PORT` : 用来连接 WebSphere MQ 队列管理者的端口
 - `MQ.CLIENT` : 传输类型
- 对于转出连接，您也必须熟悉下列配置属性：
 - `MQ.CONNECTIONFACTORY.NAME` : 提供连接给远程系统的连接工厂实例的名称



注意

下面是 IBM 提供的默认配置，您可以根据需要进行修改。更多信息请参考 WebSphere MQ 文档。

过程 24.8. 手动部署资源适配器

1. 如果您需要 WebSphereMQ 资源适配器支持事务，您必须重新打包 `wmq.jmsra-VERSION.rar` 归档，使其包含 `mqetclient.jar`。您可以使用下列命令：

```
[user@host ~]$ jar -uf wmq.jmsra-VERSION.rar mqetclient.jar
```

请用正确的版本号码替换 `VERSION`。

2. 复制 `wmq.jmsra-VERSION.rar` 文件到 `EAP_HOME/standalone/deployments/` 目录。
3. 在服务器配置文件里添加资源适配器。
 - a. 在编辑器里打开 `EAP_HOME/standalone/configuration/standalone-full.xml` 文件。
 - b. 在配置文件里找到 `urn:jboss:domain:resource-adapters` 子系统。
 - c. 如果没有为这个子系统定义资源适配器，请首先替换：

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1"/>
```

使用下列内容：

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">
  <resource-adapters>
    <!-- <resource-adapter> configuration listed below -->
  </resource-adapters>
</subsystem>
```

- d. 资源适配器的配置依赖您是否需要事务支持和恢复。如果您不需要事务支持，请选择下面的第一个配置步骤。如果您需要事务支持，请选择第二个配置步骤。

- 对于非事务性部署，请用下列内容替换 **<!-- <resource-adapter> configuration listed below -->**：

```
<resource-adapter>
  <archive>
    wmq.jmsra-VERSION.rar
  </archive>
  <transaction-support>NoTransaction</transaction-support>
  <connection-definitions>
    <connection-definition
      class-
name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryIm
pl"
      jndi-
name="java:jboss/MQ.CONNECTIONFACTORY.NAME"
      pool-name="MQ.CONNECTIONFACTORY.NAME">
      <config-property name="hostName">
        MQ.HOST.NAME
      </config-property>
      <config-property name="port">
        MQ.PORT
      </config-property>
      <config-property name="channel">
        MQ.CHANNEL.NAME
      </config-property>
      <config-property name="transportType">
        MQ.CLIENT
      </config-property>
      <config-property name="queueManager">
        MQ.QUEUE.MANAGER
      </config-property>
      <security>
        <security-domain>MySecurityDomain</security-
domain>
      </security>
    </connection-definition>
  </connection-definitions>
  <admin-objects>
    <admin-object
      class-
name="com.ibm.mq.connector.outbound.MQQueueProxy"
      jndi-name="java:jboss/MQ.QUEUE.NAME"
      pool-name="MQ.QUEUE.NAME">
      <config-property name="baseQueueName">
```

```

        MQ.QUEUE.NAME
      </config-property>
      <config-property name="baseQueueManagerName">
        MQ.QUEUE.MANAGER
      </config-property>
      <admin-object class-
name="com.ibm.mq.connector.outbound.MQTopicProxy"
        jndi-name="java:jboss/MQ.TOPIC.NAME" pool-
name="MQ.TOPIC.NAME">
        <config-property name="baseTopicName">
          MQ.TOPIC.NAME
        </config-property>
        <config-property name="brokerPubQueueManager">
          MQ.QUEUE.MANAGER
        </config-property>
      </admin-object>
    </admin-object>
  </admin-objects>
</resource-adapter>

```

请确保用 RAR 名称里的实际版本替换 *VERSION*。

- 对于事务性部署，请用下列内容替换 `<!-- <resource-adapter> configuration listed below -->`：

```

<resource-adapter>
  <archive>
    wmq.jmsra-VERSION.rar
  </archive>
  <transaction-support>XATransaction</transaction-support>
  <connection-definitions>
    <connection-definition
      class-
name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryIm
pl"
      jndi-
name="java:jboss/MQ.CONNECTIONFACTORY.NAME"
      pool-name="MQ.CONNECTIONFACTORY.NAME">
      <config-property name="hostName">
        MQ.HOST.NAME
      </config-property>
      <config-property name="port">
        MQ.PORT
      </config-property>
      <config-property name="channel">
        MQ.CHANNEL.NAME
      </config-property>
      <config-property name="transportType">
        MQ.CLIENT
      </config-property>
      <config-property name="queueManager">
        MQ.QUEUE.MANAGER
      </config-property>
    <security>
      <security-domain>MySecurityDomain</security-
domain>

```

```

        </security>
        <recovery>
            <recover-credential>
                <user-name>USER_NAME</user-name>
                <password>PASSWORD</password>
            </recover-credential>
        </recovery>
    </connection-definition>
</connection-definitions>
<admin-objects>
    <admin-object
        class-
name="com.ibm.mq.connector.outbound.MQQueueProxy"
        jndi-name="java:jboss/MQ.QUEUE.NAME"
        pool-name="MQ.QUEUE.NAME">
        <config-property name="baseQueueName">
            MQ.QUEUE.NAME
        </config-property>
        <config-property name="baseQueueManagerName">
            MQ.QUEUE.MANAGER
        </config-property>
    </admin-object>
    <admin-object class-
name="com.ibm.mq.connector.outbound.MQTopicProxy"
        jndi-name="java:jboss/MQ.TOPIC.NAME" pool-
name="MQ.TOPIC.NAME">
        <config-property name="baseTopicName">
            MQ.TOPIC.NAME
        </config-property>
        <config-property name="brokerPubQueueManager">
            MQ.QUEUE.MANAGER
        </config-property>
    </admin-object>
</admin-objects>
</resource-adapter>

```

请确保用 RAR 名称里的实际版本替换 **VERSION**。您也必须用有效的用户名和密码替换 **USER_NAME** 和 **PASSWORD**。



注意

为了支持事务，**<transaction-support>** 元素要设置为 **XATransaction**。为了支持 XA 恢复，**<recovery>** 元素要添加至连接定义里。

- e. 如果您想将 JBoss EAP 6 里的默认 EJB3 消息系统 HornetQ 修改为 WebSphere MQ，请这样修改 **urn:jboss:domain:ejb3:1.2** 子系统：

替换：

```

<mdb>
    <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
    <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>

```

为：

```
<mdb>
  <resource-adapter-ref resource-adapter-name="wmq.jmsra-
VERSION.rar"/>
  <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>
```

请确保用 RAR 名称里的实际版本替换 *VERSION*。

过程 24.9. 修改 MDB 代码来使用资源适配器

- 在 MDB 代码里这样配置 `ActivationConfigProperty` 和 `ResourceAdapter`

```
@MessageDriven( name="WebSphereMQMDB",
    activationConfig =
    {
        @ActivationConfigProperty(propertyName =
"destinationType",propertyValue = "javax.jms.Queue"),
        @ActivationConfigProperty(propertyName = "useJNDI",
propertyValue = "false"),
        @ActivationConfigProperty(propertyName = "hostName",
propertyValue = "MQ.HOST.NAME"),
        @ActivationConfigProperty(propertyName = "port",
propertyValue = "MQ.PORT"),
        @ActivationConfigProperty(propertyName = "channel",
propertyValue = "MQ.CHANNEL.NAME"),
        @ActivationConfigProperty(propertyName = "queueManager",
propertyValue = "MQ.QUEUE.MANAGER"),
        @ActivationConfigProperty(propertyName = "destination",
propertyValue = "MQ.QUEUE.NAME"),
        @ActivationConfigProperty(propertyName = "transportType",
propertyValue = "MQ.CLIENT")
    })
@ResourceAdapter(value = "wmq.jmsra-VERSION.rar")
@TransactionAttribute(TransactionAttributeType.NOT_SUPPORTED)
public class WebSphereMQMDB implements MessageListener {
}
```

请确保用 RAR 名称里的实际版本替换 *VERSION*。

[提交 bug 报告](#)

24.9. 安装 JBOSS ACTIVE MQ 资源适配器

执行下列步骤来安装 JBoss Active MQ (A-MQ) 资源适配器，使其和 JBoss Fuse A-MQ 6.1.0 服务器一起工作：https://access.redhat.com/site/documentation/en-US/Red_Hat_JBoss_Fuse/6.1/html/Deploying_into_a_Web_Server/files/DeployRar.html。

[提交 bug 报告](#)

24.10. 配置通用的 JMS 资源适配器以用于第三方的 JMS 供应商

概述

您可以配置 JBoss EAP 6 和第三方的 JMS 供应商一起使用，然而不是所有的 JMS 供应商都为集成 Java 应用程序而编写 JMS JCA 资源适配器。这个过程涵盖了配置 JBoss EAP 6 里包含的通用 JMS 资源适配器来连接 JMS 供应商的步骤。在这个过程中，Tibco EMS 6.3 用作示例 JMS 供应商，其他的 JMS 供应商需要不同的配置。



重要

通用 JMS JCA 资源适配器只应该在 JMS 供应商没有提供自己的资源适配器时才使用。在使用通用 JMS 资源适配器之前，您应该首先检查 JMS 供应商是否有可用于 JBoss EAP 6 的自己的资源适配器。

必须具备的条件

这个过程假设 JMS 供应商服务器已配置好并已准备好运行。您需要准备这个供应商的 JMS 实现所需的任何二进制文件。您也需要知道下列 JMS 供应商属性的值：

- *PROVIDER_HOST:PROVIDER_PORT* : JMS 供应商服务器的主机名和端口号。
- *PROVIDER_CONNECTION_FACTORY* : 部署在 JMS 供应商服务器上的连接工厂的名称。它必须为 XA。
- *PROVIDER_QUEUE, PROVIDER_TOPIC* : 要使用的 JMS 供应商服务器的队列和主题的名称。

过程 24.10. 配置通用 JMS 资源适配器

1. 为队列和主题的 JNDI 绑定创建一个 **ObjectFactory** 实现：
 - a. 将下列内容作为模版，用您自己的 JMS 供应商服务器的值替换服务器细节。

```
import java.util.Hashtable;
import java.util.Properties;

public class RemoteJMSObjectFactory implements ObjectFactory {

    private Context context = null;

    public RemoteJMSObjectFactory() {
    }

    public Object getObjectInstance(Object obj, Name name, Context
nameCtx,
        Hashtable<?, ?> environment) throws Exception {
        try {
            String jndi = (String) obj;

            final Properties env = new Properties();
            env.put(Context.INITIAL_CONTEXT_FACTORY,
                "com.tibco.tibjms.naming.TibjmsInitialContextFactory");
            env.put(Context.URL_PKG_PREFIXES,
                "com.tibco.tibjms.naming");
            env.put(Context.PROVIDER_URL,
                "tcp://TIBCO_HOST:TIBCO_PORT");

            context = new InitialContext(env);
            Object o = context.lookup(jndi);
```



```

        return o;
    } catch (NamingException e) {
        e.printStackTrace();
        throw e;
    }
}
}
}

```

b. 编译上面的代码，并将生成的类文件保存在 **remoteJMSObjectFactory.jar** 里。

2. 为你的 JBoss EAP 6 实例创建一个 **genericjms** 模块：

a. 创建下列目录结

构：**EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main**

b. 复制 **remoteJMSObjectFactory.jar** 文件到

EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main。

c. Copy the binaries required for the provider's JMS implementation to

EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main. For Tibco EMS, the binaries required are **tibjms.jar** and **tibcrypt.jar** from the Tibco installation's **/lib** directory.

d. 在

EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main 里创建一个 **module.xml** 文件，将之前步骤里的 JAR 文件列为资源：

```

<module xmlns="urn:jboss:module:1.1"
name="org.jboss.genericjms.provider">
  <resources>
    <resource-root path="tibjms.jar"/>
    <resource-root path="tibcrypt.jar"/>
    <resource-root path="remoteJMSObjectFactory.jar"/>
  </resources>

  <dependencies>
    <module name="javax.api"/>
    <module name="javax.jms.api"/>
  </dependencies>
</module>

```

3. 通过全局模块的方式将通用 JMS 模块添加为所有部署的依赖关系。



注意

在这个过程里，**EAP_HOME/standalone/configuration/standalone-full.xml** 被用作 JBoss EAP 6 配置文件。

在 **EAP_HOME/standalone/configuration/standalone-full.xml** 的 **<subsystem xmlns="urn:jboss:domain:ee:1.1">** 下添加：


```

<global-modules>
  <module name="org.jboss.genericjms.provider" slot="main"/>
  <module name="org.jboss.common-core" slot="main"/>
</global-modules>

```

4. 用通用的资源适配器替换默认的 HornetQ 资源适配器。

在 **EAP_HOME/standalone/configuration/standalone-full.xml** 里，下列内容替换 **<subsystem xmlns="urn:jboss:domain:ejb3:1.4"> <mdb>**：

```

<mdb>
  <resource-adapter-ref resource-adapter-
name="org.jboss.genericjms"/>
  <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>

```

5. 因为远程对象是必需的，请为您的 JMS 主题和队列添加绑定。

在 **EAP_HOME/standalone/configuration/standalone-full.xml** 里的 **<subsystem xmlns="urn:jboss:domain:naming:1.3">** 下，添加绑定，并按需要替换 **PROVIDER_QUEUE** 和 **PROVIDER_TOPIC**：

```

<bindings>
  <object-factory name="PROVIDER_QUEUE"
module="org.jboss.genericjms.provider"
class="org.jboss.qa.RemoteJMSObjectFactory"/>
  <object-factory name="PROVIDER_TOPIC"
module="org.jboss.genericjms.provider"
class="org.jboss.qa.RemoteJMSObjectFactory"/>
</bindings>

```

6. 在 **EAP_HOME/standalone/configuration/standalone-full.xml** 里，添加通用资源适配器配置到 **<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">**。

用 JMS 供应商的值替换 **PROVIDER_CONNECTION_FACTORY**、**PROVIDER_HOST** 和 **PROVIDER_PORT**。

```

<resource-adapters>
  <resource-adapter id="org.jboss.genericjms">
    <module slot="main" id="org.jboss.genericjms"/>
    <transaction-support>NoTransaction</transaction-support>
    <connection-definitions>
      <connection-definition class-
name="org.jboss.resource.adapter.jms.JmsManagedConnectionFactory"
jndi-name="java:/jms/PROVIDER_CONNECTION_FACTORY" pool-
name="PROVIDER_CONNECTION_FACTORY">
        <config-property name="JndiParameters">

java.naming.factory.initial=com.tibco.tibjms.naming.TibjmsInitialCon
textFactory;java.naming.provider.url=tcp://PROVIDER_HOST:PROVIDER_PO
RT

        </config-property>
      <config-property name="ConnectionFactory">
        PROVIDER_CONNECTION_FACTORY
      </config-property>
    </connection-definitions>
  </resource-adapter>
</resource-adapters>

```

```

        </config-property>
        <security>
            <application/>
        </security>
    </connection-definition>
</connection-definitions>
</resource-adapter>
</resource-adapters>

```

结果

配置了通用的 JMS 资源适配器以供使用。

当创建新的 **Message Driven Bean (MDB)** 时，请用类似下面的代码来使用资源适配器。用 JMS 供应商的值来替换 **PROVIDER_CONNECTION_FACTORY**、**PROVIDER_HOST** 和 **PROVIDER_PORT**。

作为可选项，下面的例子也通过 **user** 和 **password** 属性（按需要替换相应的值）配置了用于 Tibco EMS 的安全连接。

```

@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName = "destinationType",
        propertyValue = "javax.jms.Queue"),
    @ActivationConfigProperty(propertyName = "jndiParameters", propertyValue =
        "java.naming.factory.initial=com.tibco.tibjms.naming.TibjmsInitialContextF
        actory;java.naming.provider.url=tcp://PROVIDER_HOST:PROVIDER_PORT")
    @ActivationConfigProperty(propertyName = "destination", propertyValue =
        "PROVIDER_QUEUE"),
    @ActivationConfigProperty(propertyName = "connectionFactory",
        propertyValue = "PROVIDER_CONNECTION_FACTORY"),
    @ActivationConfigProperty(propertyName = "user", propertyValue =
        "USER"),
    @ActivationConfigProperty(propertyName = "password", propertyValue =
        "PASSWORD"),
})

@ResourceAdapter("org.jboss.genericjms")
public class SampleMdb implements MessageListener {
    @Override

    public void onMessage(Message message) {

    }

}

```

[提交 bug 报告](#)

第 25 章 在 AMAZON EC2 上部署 JBOSS EAP 6

25.1. 介绍

25.1.1. 关于 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) 是 amazon.com 运行的一个服务，它为客户提供可定制的虚拟计算环境。Amazon Machine Image (AMI) 可以用这个服务引导来创建虚拟机或实例。用户可以在这个实例上安装任何他需要的软件，并根据使用情况来收费。Amazon EC2 的目的是允许用户快速扩充部署的应用程序及提供灵活性。

您可以在 Amazon EC2 的网站上了解详情：<http://aws.amazon.com/ec2/>。

[提交 bug 报告](#)

25.1.2. 关于 Amazon Machine Instance (AMI)

Amazon Machine Image (AMI) 是一个用于 EC2 虚拟机实例的模版。用户通过选择合适的 AMI 来创建 EC2 实例。AMI 的主要组件是一个只读的文件系统，它包含一个已安装的操作系统及其他软件。每个 AMI 都为不同的用例安装了不同的软件。Amazon EC2 包含了 amazon.com 和第三方提供的许多 AMI。用户也可以创建自己的 AMI。

[提交 bug 报告](#)

25.1.3. 关于 JBoss Cloud Access

JBoss Cloud Access 是一个红帽的订阅功能，它提供对红帽认证的云基础结构供应商如 Amazon EC2 的支持。JBoss Cloud Access 允许您在传统服务器和基于云的公共资源以简单和高性价比的方式移动订阅。

您可以在这里找到更多细节：<http://www.redhat.com/solutions/cloud/access/jboss/>。

[提交 bug 报告](#)

25.1.4. JBoss Cloud Access 的功能

JBoss Cloud Access 计划的会员提过对 Red Hat 创建的私有 Amazon Machine Images (AMIs) 的访问。

Red Hat AMI 有下列预安装的软件且由 Red Hat 完全支持：

- Red Hat Enterprise Linux 6
- JBoss EAP 6
- JBoss Operations Network (JON) 3 代理
- 使用 Red Hat Update 基础结构的 RPM 产品更新。

每个 Red Hat AMI 都只是一个起点，要求进一步的配置来满足应用程序的需要。



重要

JBoss Cloud Access 目前不支持 full-ha 配置集，无论是独立实例还是受管域。

[提交 bug 报告](#)

25.1.5. 支持的 Amazon EC2 实例类型

JBoss Cloud Access 支持下列 Amazon EC2 实例类型。关于每种实例类型的详情，请参考《Amazon EC2 用户指南》：<http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/instance-types.html>。

表 25.1. 支持的 Amazon EC2 实例类型

实例类型	描述
标准实例	标准实例（Standard Instance）是具有均衡的内存-CPU 比例的普通环境。
高内存实例	高内存实例（High Memory Instance）比标准实例分配了更多的内存。高内存实例适用于高吞吐量的应用程序，如数据库或用内存进行缓存的应用程序。
高 CPU 实例	高 CPU 实例（High CPU Instance）比高内存实例分配了更多的 CPU 资源，它适合较低的吞吐量但高度使用 CPU 的应用程序。



重要

实例类型 **Micro** (**t1.micro**) 不适合于 JBoss EAP 6 的部署。

[提交 bug 报告](#)

25.1.6. 受支持的 Red Hat AMI

受支持的 Red Hat AMI 可以用 AMI 名称来标识。

JBoss EAP 6 AMI 使用下列语法来命名：

```
RHEL-osversion-JBEAP-6.0.0-arch-creationdate
```

osversion 是安装在 AMI 里的 Red Hat 企业版 Linux 的版本。例如：**6.2**。

arch AMI 的架构。它是 **x86_64** 或 **i386**。

creationdate AMI 创建的日期，格式为 **YYYYMMDD**。例如 **20120501**。

AMI 名称示例：**RHEL-6.2-JBEAP-6.0.0-x86_64-20120501**。

[提交 bug 报告](#)

25.2. 在 AMAZON EC2 上部署 JBOSS EAP 6

25.2.1. 在 Amazon EC2 上部署 JBoss EAP 6

JBoss EAP 6 可以用 Amazon EC2 AMI 进行部署。AMI 包含群集和非群集示例部署所需的一切元素。

部署非群集实例是最简单的情况。它只要求您在创建实例时修改一些配置来指定应用程序的部署。

部署群集实例复杂得多。除了群集实例外，您需要部署 JBoss EAP 6 实例来充当 `mod_cluster` 代理以及 S3 Bucker 用于 S3_PING JGroups 发现协议。Red Hat 也推荐创建一个虚拟私有云来容纳您的群集。

下面详述了每个步骤，但也假定您具有相关的 JBoss EAP 6、Red Hat Enterprise Linux 6 和 Amazon EC2 的经验。

我们推荐参考下列文档：

- JBoss EAP 6 - https://access.redhat.com/site/documentation/JBoss_Enterprise_Application_Platform/。
- Red Hat 企业版 Linux 6 - https://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/。
- Amazon Web Services - <http://aws.amazon.com/documentation/>。

[提交 bug 报告](#)

25.2.2. 非群集的 JBoss EAP 6

25.2.2.1. 关于非群集的实例

非群集实例是运行 JBoss EAP 6 的单个 Amazon EC2 实例。它不是群集的一部分。

[提交 bug 报告](#)

25.2.2.2. 非群集实例

25.2.2.2.1. 启动非群集的 JBoss EAP 6 实例

概述

本节涵盖在 Red Hat AMI (Amazon Machine Image) 上启动 JBoss EAP 6 非群集实例的步骤。

必须具备的条件

- 合适的 Red Hat AMI。请参考 [第 25.1.6 节“受支持的 Red Hat AMI”](#)。
- 预配置的安全组，它至少允许端口 22、8080 和 9990 上的转入请求。

过程 25.1. 在 Red Hat AMI (Amazon Machine Image) 上启动 JBoss EAP 6 非群集实例。

1. 配置 **User Data** 字段。可配置的参数位于：[第 25.4.1 节“永久性的配置参数”](#)、[第 25.4.2 节“自定义脚本参数”](#)。

例 25.1. 『User Data』字段示例

这个例子展示了非群集 JBoss EAP 6 实例的“User Data”字段。用户 `admin` 的密码已被设置为 `adminpwd`。

```
JBOSAS_ADMIN_PASSWORD=adminpwd
JBOSS_IP=0.0.0.0 #listen on all IPs and interfaces

# In production, access to these ports needs to be restricted for
security reasons
```

```

PORTS_ALLOWED="9990 9443"

cat> $USER_SCRIPT << "EOF"

# Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app
name>.war -O /usr/share/java/jboss-ec2-eap-applications/<app
name>.war

# Create a file of CLI commands to be executed after starting the
server
cat> $USER_CLI_COMMANDS << "EOC"
# deploy /usr/share/java/jboss-ec2-eap-applications/<app name>.war
EOC

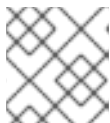
EOF

```

2. 对于产品实例

对于产品实例，在 **User Data** 字段的 **USER_SCRIPT** 行下面添加下列内容，确保在引导时进行安全更新。

```
yum -y update
```



注意

yum -y update 应该经常运行以应用安全修复和增强。

3. 启动 Red Hat AMI 实例。

结果

已在 Red Hat AMI 上配置并启动 JBoss EAP 6 的非群集实例。

[提交 bug 报告](#)

25.2.2.2.2. 在非群集 JBoss EAP 6 实例上部署应用程序

概述

本节涵盖在 Red Hat AMI 上的非群集 JBoss EAP 6 实例上部署应用程序。

1. 部署示例应用程序

在 **User Data** 字段添加下列行：

```

# Deploy the sample application from the local filesystem
deploy --force /usr/share/java/jboss-ec2-eap-samples/hello.war

```

例 25.2. 例程的「User Data」字段示例

这个例子使用了 Red Hat AMI 提供的例程。它也包含了非群集 JBoss EAP 6 实例的基本配置。用户 **admin** 的密码已被设置为 **adminpwd**。

```

JBOSAS_ADMIN_PASSWORD=adminpwd
JBOS_IP=0.0.0.0 #listen on all IPs and interfaces

# In production, access to these ports needs to be restricted
for security reasons
PORTS_ALLOWED="9990 9443"

cat> $USER_SCRIPT << "EOF"

# Create a file of CLI commands to be executed after starting
the server
cat> $USER_CLI_COMMANDS << "EOC"

# Deploy the sample application from the local filesystem
deploy --force /usr/share/java/jboss-ec2-eap-samples/hello.war
EOC

EOF

```

部署自定义的应用程序

在 **User Data** 字段添加下列行，配置应用程序名称和 URL：

```

# Get the application to be deployed from an Internet URL
mkdir -p /usr/share/java/jboss-ec2-eap-applications
wget https://<your secure storage hostname>/<path>/<app name>.war
-O /usr/share/java/jboss-ec2-eap-applications/<app name>.war

```

例 25.3. 自定义应用程序的『User Data』字段示例

这个例子使用了名为 **MyApp** 的应用程序，它也包含了非群集 JBoss EAP 6 实例的基本配置。用户 **admin** 的密码已被设置为 **adminpwd**。

```

JBOSAS_ADMIN_PASSWORD=adminpwd
JBOS_IP=0.0.0.0 #listen on all IPs and interfaces

# In production, access to these ports needs to be restricted
for security reasons
PORTS_ALLOWED="9990 9443"

cat> $USER_SCRIPT << "EOF"

# Get the application to be deployed from an Internet URL
mkdir -p /usr/share/java/jboss-ec2-eap-applications
wget https://PATH_TO_MYAPP/MyApp.war -O /usr/share/java/jboss-ec2-eap-applications/MyApp.war

# Create a file of CLI commands to be executed after starting
the server
cat> $USER_CLI_COMMANDS << "EOC"
deploy /usr/share/java/jboss-ec2-eap-applications/MyApp.war
EOC

```



2. 启动 Red Hat AMI 实例。

结果

应用程序已成功部署到 JBoss EAP 6 里。

[提交 bug 报告](#)

25.2.2.2.3. 测试非群集 JBoss EAP 6 实例

概述

本节涵盖了确认非群集 JBoss EAP 6 实例正确运行的步骤。

过程 25.2. 测试非群集 JBoss EAP 6 实例是否正确运行

1. 确定实例的细节面板里的 **Public DNS**。
2. 进入 `http://<public-DNS>:8080`。
3. 确认 JBoss EAP 主页出现并包括到管理控制台的链接。如果主页不可用，请参考：[第 25.5.1 节“关于 Amazon EC2 的故障解除”](#)。
4. 点击 **Admin Console** 链接。
5. 登录：
 - 用户名：**admin**
 - 密码：在 **User Data** 字段指定的：[第 25.2.2.2.1 节“启动非群集的 JBoss EAP 6 实例”](#)。
6. 测试例程
访问 `http://<public-DNS>:8080/hello` 来测试例程是否已正确运行。文本 **Hello World!** 应该出现在浏览器里。如果没看到这个文本，请参考：[第 25.5.1 节“关于 Amazon EC2 的故障解除”](#)。
7. 登出 JBoss EAP 6 管理控制台。

结果

JBoss EAP 6 实例已正确运行。

[提交 bug 报告](#)

25.2.2.3. 非群集的受管域

25.2.2.3.1. 启动一个实例作为域控制器

概述

本节涵盖在 Red Hat AMI (Amazon Machine Image) 上启动非群集 JBoss EAP 6 受管域的步骤。

必须具备的条件

- 合适的 Red Hat AMI。请参考 [第 25.1.6 节“受支持的 Red Hat AMI”](#)。
- [第 25.2.3.4 节“创建虚拟私有云 \(VPC\)”](#)
- [第 25.2.3.5 节“启动 Apache HTTP 服务器实例作为 VPC 的 mod_cluster 代理和 NAT 实例”](#)
- [第 25.2.3.6 节“配置 VPC 私有子网的默认路由”](#)
- [第 25.2.3.8 节“配置 IAM 设置”](#)
- [第 25.2.3.10 节“配置 S3 Bucket”](#)

过程 25.3. 在 Red Hat AMI (Amazon Machine Image) 上启动非群集 JBoss EAP 6 受管域。

1. 在 Security Group 标签页里，允许所有流量。如果需要的话用 Red Hat 企业版 Linux 的内置防火墙功能来限制访问。
2. 设置 VPC 的公共子网为 *running*。
3. 选择一个静态 IP 地址。
4. 配置 **User Data** 字段。可配置的参数位于：[第 25.4.1 节“永久性的配置参数”](#)、[第 25.4.2 节“自定义脚本参数”](#)。关于 Amazon EC2 上域控制器发现的进一步信息，请参考 [第 25.2.2.3.4 节“配置 Amazon EC2 上的域控制器发现和失效切换”](#)。

例 25.4. 『User Data』 字段示例

这个例子展示了非群集 JBoss EAP 6 受管域的“User Data”字段。用户 **admin** 的密码已被设置为 **adminpwd**。

```
## password that will be used by slave host controllers to connect
to the domain controller
JBOSSAS_ADMIN_PASSWORD=admin

## subnet prefix this machine is connected to
SUBNET=10.0.0.

## S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

##### to run the example no modifications below should be needed
#####
JBOSS_DOMAIN_CONTROLLER=true
PORTS_ALLOWED="9999 9990 9443"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on
public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app
name>.war -O /usr/share/java/jboss-ec2-eap-applications/<app
```

```

name>.war

## Create a file of CLI commands to be executed after starting the
server
cat> $USER_CLI_COMMANDS << "EOC"

# Add the modcluster subsystem to the default profile to set up a
proxy
/profile=default/subsystem=web/connector=ajp:add(name=ajp,protocol
=AJP/1.3,scheme=http,socket-binding=ajp)
/:composite(steps=[ {"operation" => "add", "address" => [
("profile" => "default"), ("subsystem" => "modcluster") ] },{
"operation" => "add", "address" => [ ("profile" => "default"),
("subsystem" => "modcluster"), ("mod-cluster-config" =>
"configuration") ], "advertise" => "false", "proxy-list" =>
"${jboss.modcluster.proxyList}", "connector" => "ajp"}, {
"operation" => "add", "address" => [ ("profile" => "default"),
("subsystem" => "modcluster"), ("mod-cluster-config" =>
"configuration"), ("dynamic-load-provider" => "configuration") ]},
{ "operation" => "add", "address" => [ ("profile" => "default"),
("subsystem" => "modcluster"), ("mod-cluster-config" =>
"configuration"), ("dynamic-load-provider" => "configuration"),
("load-metric" => "busyness")], "type"=> "busyness"} ])

# Deploy the sample application from the local filesystem
deploy /usr/share/java/jboss-ec2-eap-samples/hello.war --server-
groups=main-server-group
EOC

## this will workaround the problem that in a VPC, instance
hostnames are not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e ":::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
    echo -e "$SUBNET$i\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF

```

5. 对于产品实例

对于产品实例，在 **User Data** 字段的 **USER_SCRIPT** 行下面添加下列内容，确保在引导时进行安全更新。

```
yum -y update
```



注意

yum -y update 应该经常运行以应用安全修复和增强。

6. 启动 Red Hat AMI 实例。

结果

已在 Red Hat AMI 上配置并启动非群集的 JBoss EAP 6 受管域。

[提交 bug 报告](#)

25.2.2.3.2. 启动一个或多个实例来充当主机控制器

概述

本节涵盖了启动一个或多个 JBoss EAP 6 实例作为 Red Hat AMI（Amazon Machine Image）上的非群集主机控制器的步骤。

必须具备的条件

- 配置和启动非群集域控制器。请参考 [第 25.2.2.3.1 节“启动一个实例作为域控制器”](#)。
- [第 25.2.3.8 节“配置 IAM 设置”](#)
- [第 25.2.3.10 节“配置 S3 Bucket”](#)

过程 25.4. 启动主机控制器

对您要创建的每个实例，重复下列步骤：

1. 选择 AMI
2. 定义所需的实例数量（从主机控制器数量）。
3. 选择 VPC 和实例类型。
4. 点击『安全组』。
5. 确保来自 JBoss EAP 6 子网的所有流量都是被允许的。
6. 按需要定义其他限制。
7. 在『User Data』字段添加下列内容：

```
## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## host controller setup
### static domain controller discovery setup
JBOSS_DOMAIN_MASTER_ADDRESS=10.0.0.5
### S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

JBOSS_HOST_PASSWORD=<password for slave host controllers>

## subnet prefix this machine is connected to
SUBNET=10.0.1.

##### to run the example no modifications below should be needed #####
JBOSS_HOST_USERNAME=admin
PORTS_ALLOWED="1024:65535"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on
```

```

public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Server instance configuration
sed -i "s/other-server-group/main-server-group/"
$JBoss_CONFIG_DIR/$JBoss_HOST_CONFIG

## this will workaround the problem that in a VPC, instance
hostnames are not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e ":::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
    echo -e "$SUBNET$i\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF

```

关于 Amazon EC2 上的域控制器发现的更多信息，请参考 [第 25.2.2.3.4 节“配置 Amazon EC2 上的域控制器发现和失效切换”](#)。

8. 对于产品实例

对于产品实例，在 **User Data** 字段的 **USER_SCRIPT** 行下面添加下列内容，确保在引导时进行安全更新。

```
yum -y update
```



注意

yum -y update 应该经常运行以应用安全修复和增强。

9. 启动 Red Hat AMI 实例。

结果

已在 Red Hat AMI 上配置并启动了 JBoss EAP 6 非群集主机控制器。

[提交 bug 报告](#)

25.2.2.3.3. 测试非群集 JBoss EAP 6 受管域

概述

本节涵盖在 Red Hat AMI（Amazon Machine Image）上测试非群集 JBoss EAP 6 受管域的步骤。

要测试受管域，您必须知道 Apache HTTP 服务器和 JBoss EAP 6 域控制器的弹性 IP 地址。

必须具备的条件

- 配置和启动域控制器。请参考 [第 25.2.2.3.1 节“启动一个实例作为域控制器”](#)。
- 配置和启动主机控制器。请参考 [第 25.2.2.3.2 节“启动一个或多个实例来充当主机控制器”](#)。

过程 25.5. 测试 Web 服务器

- 用浏览器访问 **http://ELASTIC_IP_OF_APACHE_HTTPD** 来确认 Web 服务器已成功运行。

过程 25.6. 测试域控制器

1. 进入 `http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console`
2. 用用户名 `admin` 和 User Data 字段里为域控制器指定的密码登录。登录后，受管域的管理控制台登录页面将出现
(`http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console/App.html#server-instances`) 。
3. 点击屏幕右上角的 **Server** 标签。在屏幕左上角的 **Host** 下拉菜单里选择任何主机控制器。
4. 检验每个主机控制器有两个服务器配置：`server-one` 和 `server-two`，并检查它们都属于 `main-server-group`。
5. 登出 JBoss EAP 6 管理控制台。

过程 25.7. 测试主机控制器

1. 访问 `http://ELASTIC_IP_OF_APACHE_HTTPD/hello` 来测试例程是否成功运行。**Hello World!** 文本应该出现在浏览器页面里。

如果没看到这个文本，请参考：Section 18.5.1 - "About Troubleshooting Amazon EC2"。

2. 连接至 Apache HTTP 服务器实例：

```
$ ssh -L7654:localhost:7654 ELASTIC_IP_OF_APACHE_HTTPD
```

3. 访问 `http://localhost:7654/mod_cluster-manager` 以确认所有的实例都正常运行。

结果

Red Hat AMI 上的 JBoss EAP 6 Web 服务器、域控制器和主机控制器都正常运行。

提交 bug 报告

25.2.2.3.4. 配置 Amazon EC2 上的域控制器发现和失效切换

对于运行在 Amazon EC2 上的受管域，除了静态的域控制器发现，主机控制器可以用 Amazon S3 存储系统动态地发现域控制器。特别是，主机控制器和域控制器可以用访问 Amazon S3 Bucket 所需的信息来进行配置。

使用这个配置，当启动域控制器时，它将其联系信息写入到 Bucket 里的 S3 文件。每当主机控制器试图联系域控制器，它都会从 S3 文件获取域控制器的联系信息。

这意味着如果域控制器的联系信息有变动（例如，EC2 实例的 IP 地址在停止或启动时发现变化是很常见的），您不需要重新进行配置主机控制器。主机控制器可以从 S3 文件获取域控制器的新联系信息。

您可以通过在启动 JBoss EAP 6 实例时传

入 `JBOSS_DOMAIN_S3_ACCESS_KEY`、`JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY` 和 `JBOSS_DOMAIN_S3_BUCKET` 参数来自动启用域控制器发现。请参考 第 25.4.1 节“永久性的配置参数”里的可配置参数。或者，您可以用下列属性来手动配置域发现。

手动的域控制器发现配置是用下列属性指定的：

access-key

Amazon AWS 用户帐号的访问密钥。

secret-access-key

Amazon AWS 用户帐号的秘密访问密钥。

location

要使用的 Amazon S3 Bucket。

下面是主机控制器和域控制器配置的例子。虽然这些例子只展示了一个发现选项，但您可以配置任意数量的静态发现或 S3 发现选项。关于域发现的细节和失效切换过程的详情，请参考 [第 1.7 节“关于域控制器发现和失效切换”](#)。

例 25.5. 主机控制器配置

```
<domain-controller>
  <remote security-realm="ManagementRealm">
    <discovery-options>
      <discovery-option name="s3-discovery"
code="org.jboss.as.host.controller.discovery.S3Discovery"
module="org.jboss.as.host-controller">
        <property name="access-key" value="S3_ACCESS_KEY"/>
        <property name="secret-access-key"
value="S3_SECRET_ACCESS_KEY"/>
        <property name="location" value="S3_BUCKET_NAME"/>
      </discovery-option>
    </discovery-options>
  </remote>
</domain-controller>
```

例 25.6. 域控制器配置

```
<domain-controller>
  <local>
    <discovery-options>
      <discovery-option name="s3-discovery"
code="org.jboss.as.host.controller.discovery.S3Discovery"
module="org.jboss.as.host-controller">
        <property name="access-key" value="S3_ACCESS_KEY"/>
        <property name="secret-access-key"
value="S3_SECRET_ACCESS_KEY"/>
        <property name="location" value="S3_BUCKET_NAME"/>
      </discovery-option>
    </discovery-options>
  </local>
</domain-controller>
```

[提交 bug 报告](#)

25.2.3. 群集的 JBoss EAP 6

25.2.3.1. 关于群集实例

群集实例是运行 JBoss EAP 6 且启用了群集的 Amazon EC2 实例。运行 Apache HTTP 服务器的另外一个实例将充当群集里实例的代理。

JBoss EAP 6 AMI 包含两个用于群集实例的配置文件：**standalone-ec2-ha.xml** 和 **standalone-mod_cluster-ec2-ha.xml**。每个配置文件都提供了群集功能而无需使用多点传送，这是因为 Amazon EC2 不支持多点传送。这是通过用于群集通讯的 TCP 单点传送和用于发现协议的 S3_PING 来实现的。**standalone-mod_cluster-ec2-ha.xml** 配置也提供 mod_cluster 代理的简单注册。

类似地，**domain-ec2.xml** 配置文件提供在群集受管域里使用的两个配置集：**ec2-ha** 和 **mod_cluster-ec2-ha**。

[提交 bug 报告](#)

25.2.3.2. 创建一个关系型数据库服务的数据库实例

概述

本节介绍创建关系型数据库服务实例的步骤，以 MySQL 为例子。



警告

我们强烈推荐在产品环境里启用备份和维护功能。



重要

为访问数据库的每个应用程序都创建单独的用户/密码是不错的做法。请按照您的应用程序的需要调整其他配置选项。

过程 25.8. 创建一个关系型数据库服务的数据库实例

1. 点击 AWS 控制台里的 **RDS**。
2. 如果需要则订阅这个服务。
3. 点击 **Launch DB instance**。
4. 点击 **MySQL**。
 - a. 选择版本。例如：**5.5.12**。
 - b. 选择 **small instance**。
 - c. 确保 **Multi-AZ Deployment** 和 **Auto upgrade** 是 **off**。
 - d. 设置 **Storage** 为 **5GB**。
 - e. 定义数据库管理员的用户名和密码并点击 **Next**。
 - f. 选择随实例一起创建的数据库的名称，并点击 **Next**。

g. 如果需要，禁用备份和维护。

h. 确认设置。

结果

数据库已创建。数分钟后它将初始化并可以使用。

[提交 bug 报告](#)

25.2.3.3. 关于虚拟私有云

Amazon 虚拟私有云（Virtual Private Cloud，VPC）是 Amazon Web Service（AWS）的一个功能，它允许您在私有网络里隔离一系列 AWS 资源。这个私有网络的拓扑和配置可以按照需要进行定制。

更多信息请参考 Amazon 虚拟私有云的网站：<http://aws.amazon.com/vpc/>。

[提交 bug 报告](#)

25.2.3.4. 创建虚拟私有云（VPC）

概述

本节以外部数据库为例，介绍创建虚拟私有云所需的步骤。您的安全策略可以要求到数据库的连接加密。关于加密数据库连接的详情，请参考 Amazon 的 *RDS FAQ*。



重要

对于 JBoss EAP 6 群集设置，我们推荐使用 VPC。这是因为它极大地简化了群集节点、JON 服务器及 mod_cluster 代理间的安全连接。如不使用 VPC，这些通讯频道都需要加密和验证。

关于配置 SSL 的详细说明，请参考：[第 11.12.1 节“对 JBoss EAP 6 Web 服务器实施 SSL 加密”](#)。

1. 点击 AWS 控制台里的 VPC 标签页。
2. 如果需要则订阅这个服务。
3. 点击 **"Create new VPC"**。
4. 选择带有一个公共子网和一个私有子网的 VPC。
 - a. 设置公共子网为 **10.0.0.0/24**。
 - b. 设置私有子网为 **10.0.1.0/24**。
5. 进入 **Elastic IPs**。
6. 创建一个 mod_cluster proxy/NAT 实例使用的弹性 IP 地址。
7. 进入 **Security groups** 并创建一个安全组来允许所有流量的进出。
8. 进入 **Network ACLs**。
 - a. 创建 ACL 来允许所有流量进出。

- b. 创建 ACL 来允许所有流量只通过 TCP 端口 22、8009、8080、8443、9443、9990 和 16163 进出。

结果

成功地创建了虚拟私有云（VPC）。

[提交 bug 报告](#)

25.2.3.5. 启动 Apache HTTP 服务器实例作为 VPC 的 mod_cluster 代理和 NAT 实例

概述

本节介绍启动 Apache HTTP 服务器实例作为 VPC 的 mod_cluster 代理和 NAT 实例所需的步骤。

必须具备的条件

- [第 25.2.3.2 节“创建一个关系型数据库服务的数据库实例”](#)。
- [第 25.2.3.4 节“创建虚拟私有云（VPC）”](#)

过程 25.9. 启动 Apache HTTP 服务器实例作为 VPC 的 mod_cluster 代理和 NAT 实例

1. 为这个实例创建一个有弹性的 IP 地址。
2. 选择 AMI
3. 进入 **Security Group** 并允许所有流量（如果需要的话用 Red Hat 企业版 Linux 的内置防火墙功能来限制访问）。
4. 在 VPC 的公共子网里选择 "running"。
5. 选择一个静态 IP (如：10.0.0.4)。
6. 在 **User Data**: 字段添加下列内容：

```
JBOSSCONF=disabled

cat > $USER_SCRIPT << "EOS"

echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 0 > /proc/sys/net/ipv4/conf/eth0/rp_filter

iptables -I INPUT 4 -s 10.0.1.0/24 -p tcp --dport 7654 -j ACCEPT
iptables -I INPUT 4 -p tcp --dport 80 -j ACCEPT

iptables -I FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -I FORWARD -s 10.0.1.0/24 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 ! -s 10.0.0.4 -j MASQUERADE

# balancer module incompatible with mod_cluster
sed -i -e 's/LoadModule proxy_balancer_module/#\0/'
/etc/httpd/conf/httpd.conf

cat > /etc/httpd/conf.d/mod_cluster.conf << "EOF"
```

```

#LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
LoadModule slotmem_module modules/mod_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule advertise_module modules/mod_advertise.so

Listen 7654

# workaround JBPAPP-4557
MemManagerFile /var/cache/mod_proxy/manager

<VirtualHost *:7654>
    <Location /mod_cluster-manager>
        SetHandler mod_cluster-manager
        Order deny,allow
        Deny from all
        Allow from 127.0.0.1
    </Location>

    <Location />
        Order deny,allow
        Deny from all
        Allow from 10.
        Allow from 127.0.0.1
    </Location>

    KeepAliveTimeout 60
    MaxKeepAliveRequests 0
    ManagerBalancerName mycluster
    ServerAdvertise Off
    EnableMCPMReceive On
</VirtualHost>
EOF

echo "`hostname | sed -e 's/ip-//' -e 'y/-/./'`" `hostname`"
>> /etc/hosts

semanage port -a -t http_port_t -p tcp 7654 #add port in the apache
port list for the below to work
setsebool -P httpd_can_network_relay 1 #for mod_proxy_cluster to
work
chcon -t httpd_config_t -u system_u
/etc/httpd/conf.d/mod_cluster.conf

#### Uncomment the following line when launching a managed domain
####
# setsebool -P httpd_can_network_connect 1

service httpd start

EOS

```

7. 禁用这个实例的 Amazon EC2 云源/目的地检查，让它可以充当路由器。

- a. 右击运行的 Apache HTTP 服务器实例并选择 **"Change Source/Dest check"**。
 - b. 点击 **Yes, Disable**。
8. 为这个实例分配弹性 IP 地址。

结果

已成功启动 Apache HTTP 服务器实例。

[提交 bug 报告](#)

25.2.3.6. 配置 VPC 私有子网的默认路由

概述

本节涵盖配置 VPC 私有子网默认路由所需的步骤。JBoss EAP 6 群集节点将运行在 VPC 的私有子网里，但群集节点要求访问互联网以实现 S3 连接性。您需要设置默认路由可以通过 NAT 实例。

过程 25.10. 配置 VPC 私有子网的默认路由

1. 在 Amazon AWS 控制台里进入 Apache HTTP 服务器实例。
2. 选择 **VPC → route tables**。
3. 点击私有子网使用的路由表。
4. 在新路由字段里输入 **0.0.0.0/0**。
5. 点击 **"Select a target"**。
6. 选择 **"Enter Instance ID"**。
7. 选择运行的 Apache HTTP 服务器实例的 ID。

结果

已成功为 VPC 子网配置了默认路由。

[提交 bug 报告](#)

25.2.3.7. 关于标识符和访问管理 (Identity and Access Management , IAM)

标识符和访问管理为 AWS 资源提供了可配置的安全性。您可以配置 IAM 使用在 IAM 里创建的帐号或提供 IAM 和自己的标识符服务间的标识符联合机制。

更多信息请参考 AWS 标识符和访问管理网站：<http://aws.amazon.com/iam/>。

[提交 bug 报告](#)

25.2.3.8. 配置 IAM 设置

概述

本节涵盖为群集 JBoss EAP 6 实例设立 IAM 所需的配置步骤。**S3_PING** 协议使用 S3 Bucket 来发现其他群集成员。**JGroups 3.0.x** 版本要求 Amazon AWS 帐号及安全密钥来对 S3 服务进行验证。

因为 S3 域控制器发现使用了 S3 Bucket，它要求 Amazon AWS 帐号访问权限和密钥来通过 S3 服务的验证（类似于 JGroups 使用的 **S3_PING** 协议）。用于 S3 发现的 IAM 用户和 S3 Bucket 必须和用于群集的 IAM 用户和 S3 Bucket 不同。

在 **user-data** 字段输入主帐号凭证是有安全风险的，请将其在线存储或放在 AMI 里。要避免这种风险，您可以用 Amazon IAM 功能来创建单独的帐号，它只被赋予对单个 S3 Bucket 的访问权限。

过程 25.11. 配置 IAM 设置

1. 点击 AWS 控制台里的 IAM 标签页。
2. 点击 **users**。
3. 选择 **Create New Users**。
4. 选择名称并确保选择 **Generate an access key for each User** 选项。
5. 选择 **Download credentials** 并将其保存在安全的位置。
6. 关闭窗口。
7. 点击新创建的用户。
8. 记下 **User ARM** 的值。设置 S3 Bucket 要求这个值，其文档位于 [第 25.2.3.10 节“配置 S3 Bucket”](#)。

结果

成功地创建了 IAM 用户帐号。

[提交 bug 报告](#)

25.2.3.9. 关于 S3 Bucket

S3 Bucket 是 Amazon 简单存储系统（Amazon S3）里地基本机构存储单元。Bucket 可以存储任意数目的对象且在 Amazon S3 里必须具有唯一的名称进行标识。

更多信息请参考 Amazon S3 网站 <http://aws.amazon.com/s3/>。

[提交 bug 报告](#)

25.2.3.10. 配置 S3 Bucket

概述

这个主题涵盖了配置新 S3 Bucket 所需的步骤。

必须具备的条件

- [第 25.2.3.8 节“配置 IAM 设置”](#)。

过程 25.12. 配置 S3 Bucket

1. 在 AWS 控制台里打开 **S3** 标签页。
2. 点击 **Create Bucket**。

3. 为这个 Bucket 选择一个名称并点击 **Create**。



注意

Bucket 名称在整个 S3 里是唯一的。它的名称不能被重用。

4. 右击新的 Bucket 并选择 **Properties**。
5. 点击 **permissions** 标签页里的 **Add bucket policy**。
6. 点击 **New policy** 来打开策略创建向导。
 - a. 复制下列内容到新的策略里，用这里定义的值替换 `arn:aws:iam::055555555555:user/jbosscluster*`：第 25.2.3.8 节“配置 IAM 设置”。修改 `clusterbucket123` 的两个实例为这个过程里步骤 3 中定义的 Bucket 的名称。

```
{
  "Version": "2008-10-17",
  "Id": "Policy1312228794320",
  "Statement": [
    {
      "Sid": "Stmt1312228781799",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::055555555555:user/jbosscluster"
        ]
      },
      "Action": [
        "s3:ListBucketVersions",
        "s3:GetObjectVersion",
        "s3:ListBucket",
        "s3:PutBucketVersioning",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:GetObject",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts",
        "s3:PutObject",
        "s3:GetBucketVersioning"
      ],
      "Resource": [
        "arn:aws:s3:::clusterbucket123/*",
        "arn:aws:s3:::clusterbucket123"
      ]
    }
  ]
}
```

结果

新的 S3 Bucket 已被成功创建和配置。

[提交 bug 报告](#)

25.2.3.11. 群集实例

25.2.3.11.1. 启动群集 JBoss EAP 6 AMI

概述

本节涵盖启动群集 JBoss EAP 6 AMI 的步骤

必须具备的条件

- 第 25.2.3.2 节 “创建一个关系型数据库服务的数据库实例”。
- 第 25.2.3.4 节 “创建虚拟私有云（VPC）”。
- 第 25.2.3.5 节 “启动 Apache HTTP 服务器实例作为 VPC 的 `mod_cluster` 代理和 NAT 实例”。
- 第 25.2.3.6 节 “配置 VPC 私有子网的默认路由”。
- 第 25.2.3.8 节 “配置 IAM 设置”。
- 第 25.2.3.10 节 “配置 S3 Bucket”。



警告

在带有小于 24 位掩码的子网里或跨越多个子网运行 JBoss EAP 6 群集要求每个群集成员都对对应唯一的服务器对等 ID。

关于如何使这样的配置可靠地运行，请参考 `JBoss_CLUSTER_ID` 变量。



重要

Amazon EC2 的 `auto-scaling` 功能可能用于 JBoss EAP 6 群集节点。然而，请确保在部署前进行测试。您应该确保特定的工作负荷分摊到多个节点上，且性能满足对计划使用的实例的需要（不同实例类型接收不同比例的 EC2 云资源）。

而且，实例位置和当前网络/存储/主机/RDS 利用率都可能影响群集的性能。请用期望的实际负荷进行测试并预计意外的条件。



警告

Amazon EC2 *scale-down* 动作终止节点而不会优雅地关闭节点，某些事务可能会被中断，其他群集节点（和负载均衡）将需要时间来进行失效切换。这很可能影响到应用程序的用户体验。

我们推荐您在 `mod_cluster` 管理接口里禁用服务器直至会话完成来缩减应用程序群集规模，或者是优雅地关闭 JBoss EAP 6 实例（可以用 SSH 访问实例或使用 JON）。

测试您缩减的过程是否导致对用户体验起反作用。对于特定的工作负荷、负载均衡和设立可能需要其他的措施。

过程 25.13. 启动群集 JBoss EAP 6 AMI

1. 选择 AMI
2. 定义想要的实例的数量（群集大小）。
3. 选择 VPC 和实例类型。
4. 点击 **Security Group**。
5. 确保来自 JBoss EAP 6 群集子网的所有流量都是被允许的。
6. 按需要定义其他限制。
7. 在 **User Data** 字段里添加下列内容：

例 25.7. 『User Data』字段示例

```
## mod_cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## clustering setup
JBOSS_JGROUPS_S3_PING_SECRET_ACCESS_KEY=<your secret key>
JBOSS_JGROUPS_S3_PING_ACCESS_KEY=<your access key>
JBOSS_JGROUPS_S3_PING_BUCKET=<your bucket name>

## password to access admin console
JBOSSAS_ADMIN_PASSWORD=<your password for opening admin console>

## database credentials configuration
JAVA_OPTS="$JAVA_OPTS -
Ddb.host=instancename.something.rds.amazonaws.com -
Ddb.database=mydatabase -Ddb.user=<user> -Ddb.passwd=<pass>"

## subnet prefix this machine is connected to
SUBNET=10.0.1.

#### to run the example no modifications below should be needed
####
```

```

PORTS_ALLOWED="1024:65535"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on
public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app
name>.war -O /usr/share/java/jboss-ec2-eap-applications/<app
name>.war

## install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbossas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-*.jar
/usr/share/jbossas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbossas/modules/com/mysql/main/module.xml <<"EOM"
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
EOM

cat > $USER_CLI_COMMANDS << "EOC"
## Deploy sample application from local filesystem
deploy --force /usr/share/java/jboss-ec2-eap-samples/cluster-
demo.war

## ExampleDS configuration for MySQL database
data-source remove --name=ExampleDS
/subsystem=datasources/jdbc-driver=mysql:add(driver-
name="mysql",driver-module-name="com.mysql")
data-source add --name=ExampleDS --connection-
url="jdbc:mysql://${db.host}:3306/${db.database}" --jndi-
name=java:jboss/datasources/ExampleDS --driver-name=mysql --user-
name="${db.user}" --password="${db.passwd}"
/subsystem=datasources/data-source=ExampleDS:enable
/subsystem=datasources/data-source=ExampleDS:test-connection-in-
pool
EOC

## this will workaround the problem that in a VPC, instance
hostnames are not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e ":::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
  echo -e "$SUBNET$i\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF

```


结果

已成功配置和启动群集 JBoss EAP 6 AMI。

[提交 bug 报告](#)

25.2.3.11.2. 测试群集 JBoss EAP 6 实例

概述

本节涵盖了确认群集 JBoss EAP 6 实例正确运行的步骤。

过程 25.14. 测试群集实例

1. 用浏览器访问 http://ELASTIC_IP_OF_APACHE_HTTPD 来确认 Web 服务器已成功运行。

2. 测试群集节点

- a. 在 Web 浏览器里访问 http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/put.jsp。

- b. 检验某个群集节点是否记录如下的消息：

```
Putting date now
```

- c. 停止在之前步骤里记录消息的节点。

- d. 在 Web 浏览器里访问 http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/get.jsp。

- e. 检验显示的时间和步骤 2-a 里用 `put.jsp` PUT 的时间相同。

- f. 检验某个运行的群集节点是否记录如下的消息：

```
Getting date now
```

- g. 重启已停止的群集节点。

- h. 连接至 Apache HTTP 服务器实例：

```
ssh -L7654:localhost:7654 <ELASTIC_IP_OF_APACHE_HTTPD>
```

- i. 访问 http://localhost:7654/mod_cluster-manager 以确认所有的实例都正常运行。

结果

已测试群集的 JBoss EAP 6 实例且确认都工作正常。

[提交 bug 报告](#)

25.2.3.12. 群集的受管域

25.2.3.12.1. 启动一个实例作为群集域控制器

概述

本节涵盖在 Red Hat AMI (Amazon Machine Image) 上启动群集 JBoss EAP 6 受管域的步骤。

必须具备的条件

- 合适的 Red Hat AMI。请参考 [第 25.1.6 节“受支持的 Red Hat AMI”](#)。
- [第 25.2.3.4 节“创建虚拟私有云 \(VPC\)”](#)
- [第 25.2.3.5 节“启动 Apache HTTP 服务器实例作为 VPC 的 mod_cluster 代理和 NAT 实例”](#)
- [第 25.2.3.6 节“配置 VPC 私有子网的默认路由”](#)
- [第 25.2.3.8 节“配置 IAM 设置”](#)
- [第 25.2.3.10 节“配置 S3 Bucket”](#)

过程 25.15. 启动群集域控制器

1. 为这个实例创建一个有弹性的 IP 地址。
2. 选择 AMI
3. 进入 Security Group 并允许所有流量 (如果需要的话用 Red Hat 企业版 Linux 的内置防火墙功能来限制访问)。
4. 在 VPC 的公共子网里选择 "running"。
5. 选择一个静态 IP (如 : **10.0.0.5**) 。
6. 在『User Data』字段添加下列内容：

```
## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## password that will be used by slave host controllers to connect
to the domain controller
JBOSSAS_ADMIN_PASSWORD=<password for slave host controllers>

## subnet prefix this machine is connected to
SUBNET=10.0.0.

## S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

#### to run the example no modifications below should be needed ####
JBOSS_DOMAIN_CONTROLLER=true
PORTS_ALLOWED="9999 9990 9443"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on
public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war
```

```

-0 /usr/share/java/jboss-ec2-eap-applications/<app name>.war

## Install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbossas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-*.jar
/usr/share/jbossas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbossas/modules/com/mysql/main/module.xml <<"EOM"
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
EOM

cat > $USER_CLI_COMMANDS << "EOC"
## Deploy the sample application from the local filesystem
deploy /usr/share/java/jboss-ec2-eap-samples/cluster-demo.war --
server-groups=other-server-group

## ExampleDS configuration for MySQL database
data-source --profile=mod_cluster-ec2-ha remove --name=ExampleDS
/profile=mod_cluster-ec2-ha/subsystem=datasources/jdbc-
driver=mysql:add(driver-name="mysql",driver-module-name="com.mysql")
data-source --profile=mod_cluster-ec2-ha add --name=ExampleDS --
connection-url="jdbc:mysql://${db.host}:3306/${db.database}" --jndi-
name=java:jboss/datasources/ExampleDS --driver-name=mysql --user-
name="${db.user}" --password="${db.passwd}"
/profile=mod_cluster-ec2-ha/subsystem=datasources/data-
source=ExampleDS:enable
EOC

## this will workaround the problem that in a VPC, instance
hostnames are not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e ":::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
  echo -e "$SUBNET$i\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF

```

7. 对于产品实例

对于产品实例，在 **User Data** 字段的 **USER_SCRIPT** 行下面添加下列内容，确保在引导时进行安全更新。

```

yum -y update

```



注意

yum -y update 应该经常运行以应用安全修复和增强。

8. 启动 Red Hat AMI 实例。

结果

已在 Red Hat AMI 上配置并启动群集的 JBoss EAP 6 受管域。

[提交 bug 报告](#)

25.2.3.12.2. 启动一个或多个实例来充当群集主机控制器

概述

本节涵盖了启动一个或多个 JBoss EAP 6 实例作为 Red Hat AMI (Amazon Machine Image) 上的群集主机控制器的步骤。

必须具备的条件

- 配置和启动群集域控制器。请参考 [第 25.2.3.12.1 节“启动一个实例作为群集域控制器”](#)。
- [第 25.2.3.8 节“配置 IAM 设置”](#)
- [第 25.2.3.10 节“配置 S3 Bucket”](#)

过程 25.16. 启动主机控制器

对您要创建的每个实例，重复下列步骤：

1. 选择 AMI
2. 定义所需的实例数量（从主机控制器数量）。
3. 选择 VPC 和实例类型。
4. 点击『安全组』。
5. 确保来自 JBoss EAP 6 群集子网的所有流量都是被允许的。
6. 按需要定义其他限制。
7. 在『User Data』字段添加下列内容：

```
## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## clustering setup
JBOSS_JGROUPS_S3_PING_SECRET_ACCESS_KEY=<your secret key>
JBOSS_JGROUPS_S3_PING_ACCESS_KEY=<your access key>
JBOSS_JGROUPS_S3_PING_BUCKET=<your bucket name>

## host controller setup
### static domain controller discovery setup
JBOSS_DOMAIN_MASTER_ADDRESS=10.0.0.5
```

```

### S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

JBOSS_HOST_PASSWORD=<password for slave host controllers>

## database credentials configuration
JAVA_OPTS="$JAVA_OPTS -
Ddb.host=instancename.something.rds.amazonaws.com -
Ddb.database=mydatabase -Ddb.user=<user> -Ddb.passwd=<pass>"

## subnet prefix this machine is connected to
SUBNET=10.0.1.

##### to run the example no modifications below should be needed #####
JBOSS_HOST_USERNAME=admin
PORTS_ALLOWED="1024:65535"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on
public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Server instance configuration
sed -i "s/main-server-group/other-server-group/"
$JBOSS_CONFIG_DIR/$JBOSS_HOST_CONFIG

## install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbossas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-*.jar
/usr/share/jbossas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbossas/modules/com/mysql/main/module.xml <<"EOM"
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
EOM

## this will workaround the problem that in a VPC, instance
hostnames are not resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e ":::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
  echo -e "$SUBNET$i\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF

```

8. 对于产品实例

对于产品实例，在 **User Data** 字段的 **USER_SCRIPT** 行下面添加下列内容，确保在引导时进行安全更新。

```
yum -y update
```



注意

yum -y update 应该经常运行以应用安全修复和增强。

9. 启动 Red Hat AMI 实例。

结果

已在 Red Hat AMI 上配置并启动了 JBoss EAP 6 群集主机控制器。

[提交 bug 报告](#)

25.2.3.12.3. 测试群集 JBoss EAP 6 受管域

概述

本节涵盖在 Red Hat AMI（Amazon Machine Image）上测试群集 JBoss EAP 6 受管域的步骤。

要测试受管域，您必须知道 Apache HTTP 服务器和 JBoss EAP 6 域控制器的弹性 IP 地址。

必须具备的条件

- 配置和启动群集域控制器。请参考 [第 25.2.3.12.1 节“启动一个实例作为群集域控制器”](#)。
- 配置和启动群集主机控制器。请参考 [第 25.2.3.12.2 节“启动一个或多个实例来充当群集主机控制器”](#)。

过程 25.17. 测试 Apache HTTP 服务器实例

- 用浏览器访问 `http://ELASTIC_IP_OF_APACHE_HTTP_SERVER` 来确认 Web 服务器已成功运行。

过程 25.18. 测试域控制器

1. 访问 `http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console`
2. 用用户名 **admin** 和 User Data 字段里为域控制器指定的密码登录。登录后，受管域的管理控制台登录页面将出现
(`http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console/App.html#server-instances`) 。
3. 点击屏幕右上角的 **Server** 标签。在屏幕左上角的 **Host** 下拉菜单里选择任何主机控制器。
4. 检验这个主机控制器有两个服务器配置：**server-one** 和 **server-two**，并检查它们都属于 **other-server-group**。

过程 25.19. 测试主机控制器

1. 在浏览器里访问 `http://ELASTIC_IP_OF_APACHE_HTTP_SERVER/cluster-demo/put.jsp`。
2. 检验其中一个主机控制器记录下列日志：`Putting date now.`
3. 停止在之前步骤里记录消息的服务器实例（请参考 *Stop a Server Using the Management Console*）。
4. 用浏览器访问 `http://ELASTIC_IP_OF_APACHE_HTTP_SERVER/cluster-demo/get.jsp`。
5. 检验显示的时间和步骤 2-a 里用 `put.jsp` PUT 的时间相同。
6. 检验其中一个服务器实例记录下列日志：`Getting date now.`
7. 重启已停止的服务器实例（请参考 2.8.3 章节 - *Start a Server Using the Management Console*）
8. 连接至 Apache HTTP 服务器实例。

```
$ ssh -L7654:localhost:7654 ELASTIC_IP_OF_APACHE_HTTP_SERVER
```

9. 访问 `http://localhost:7654/mod_cluster-manager` 以确认所有的实例都正常运行。

结果

Red Hat AMI 上的 JBoss EAP 6 Web 服务器、域控制器和主机控制器都正常运行。

[提交 bug 报告](#)

25.3. 用 JBOSS OPERATIONS NETWORK (JON) 建立监控

25.3.1. 关于 AMI 监控

将商业应用程序部署到正确配置的 AMI 实例后，下一步是用 JBoss Operations Network (JON) 建立平台的监控。

JON 服务器通常位于机构网络内部，所以有必要在服务器和代理间建立一个安全的连接。最常见的方案是在两点间建立 VPN，但这会使所需的网络配置变得复杂。本章提供了在 JON 代理和 JON 服务器间启用通讯的网络配置准则。关于配置、管理和使用的更多信息，请参考 JBoss Operations Network (JON) 官方文档。

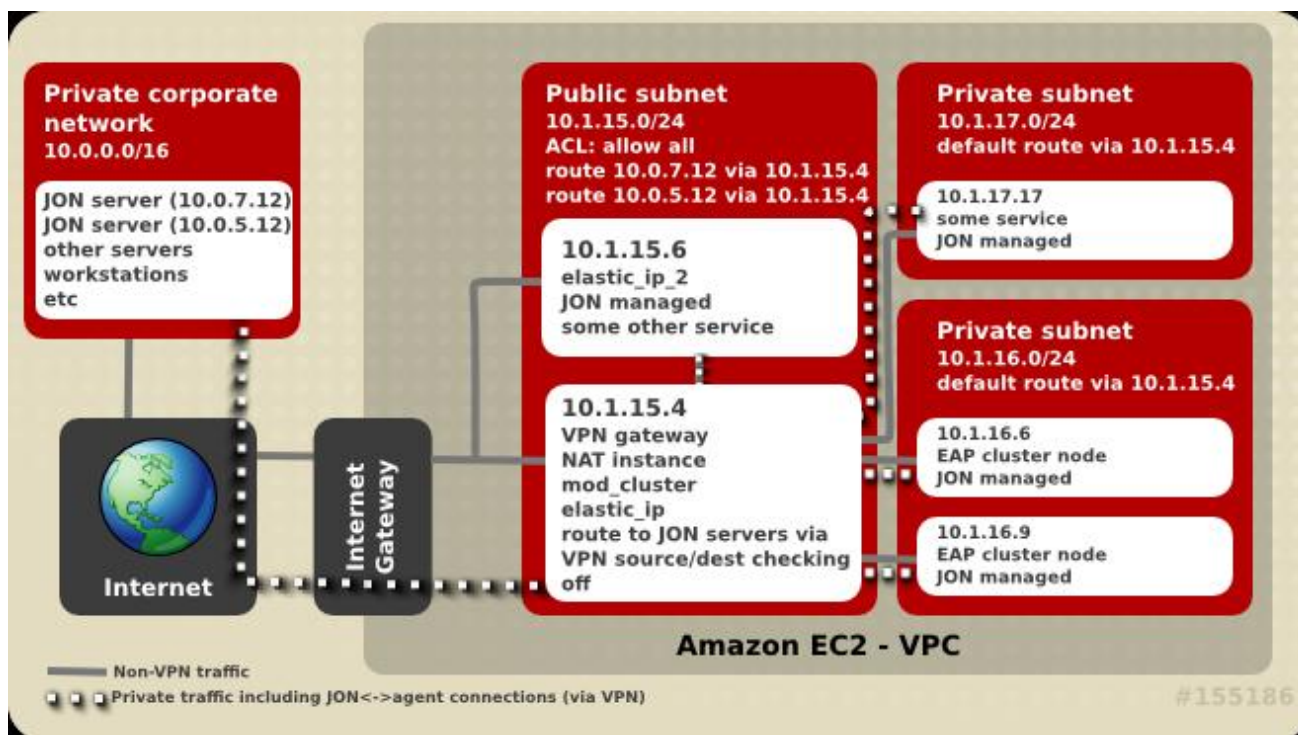


图 25.1. JON 服务器连接性

[提交 bug 报告](#)

25.3.2. 关于连接性的要求

在服务器里注册 JON 代理要求代理和服务器的双路通讯。JON 代理需要访问所有 JON 服务器上的端口 **7080**，使用 SSL 时则是端口 **7443**。每个 JON 服务器必须能够访问具有唯一主机和端口的每个连接的代理。代理端口通常是 **16163**。

如果有多个群集 JON 服务器，请确保每个代理可以通过在 JON 服务器管理控制台里配置的 IP 地址和主机名对与 JON 群集里的所有服务器进行通讯。代理注册时使用的 JON 服务器和它在初始化后试图使用的服务器可能不同。

[提交 bug 报告](#)

25.3.3. 关于网络地址转换（Network Address Translation，NAT）

路由模式的机构 VPN 网关极大地简化了网络配置。如果您的机构 VPN 网关使用 NAT 模式，JON 服务器则无法直接看到代理。此时，您必须为每个代理配置端口转发。

NAT VPN 配置要求转发网关上一个端口到受管域上 JON 代理的端口地址。您也需要配置 JON 代理以告知服务器转发端口号和 IP 地址。您可以在 `agent-configuration.xml` 配置文件的 `rhq.communications.connector.*` 描述里找到更多信息。

[提交 bug 报告](#)

25.3.4. 关于 Amazon EC2 和 DNS

JON 服务器和 JON 代理需要解析彼此的主机名。DNS 解析在 VPN 配置里更为复杂。连接的服务器有多个可能的选项。一个选项是使用 Amazon EC2 或机构网络的 DNS 服务器。另外一个选项是使用独立的 DNS 配置，机构 DNS 服务器用来解析特定域里的名称，而 Amazon EC2 DNS 服务器则用来解析所有其他的名称。

[提交 bug 报告](#)

25.3.5. 关于 EC2 里的路由

所有的 Amazon EC2 服务器默认都激活了一个 **source/destination checking** 路由功能。这个功能将发送到服务器的目的地和主机的 IP 地址不一致的数据包丢弃。如果连接代理到 JON 服务器的 VPN 方案包括了一个路由器，您需要关闭这个功能，或者将服务器作为路由器或 VPN 网关。这个配置设置可以通过 Amazon AWS 控制台进行访问。虚拟私有云（Virtual Private Cloud，VPC）里也要求禁用 **source/destination checking**。

某些 VPN 配置默认将普通的流量通过机构 VPN 进行路由。我们推荐您避免这样做，因为这可能是较慢和低效的配置方式。

虽然使用正确的地址模式并不是 JON 专有的需要担心的事情，但较差的模式可能会有不良影响。Amazon EC2 从 **10.0.0.0/8** 开始分配 IP 地址。实例通常也具有一个公用 IP 地址，但只有相同可用性区内 IP 地址上的网络流量才是自由的。要避免在私有网络里使用 **10.0.0.0/8** 段的地址，您需要考虑几件事情。

- 在创建 VPC 时，不要分配已经用在私有网络里的地址以避免连接性问题。
- 如果某个实例需要访问可用性区的本地资源，请确保使用 Amazon EC2 私有地址且流量不通过 VPN 来路由。
- 如果 Amazon EC2 实例将访问机构私有网络地址的一个小的子集（如只有 JON 服务器），只有这些地址应该通过 VPN 进行路由。这增强了安全性并降低了 Amazon EC2 或私有网络地址空间冲突的机会。

[提交 bug 报告](#)

25.3.6. 关于终止和重启 JON

云环境的其中一个益处是您可以轻易终止和启动主机实例。您也可以启动和初始实例等同的实例。如果新实例试图用和之前运行的代理相同的名称来注册 JON 服务器，那可能会出现错误。如果是这样，JON 服务器不会再允许代理用缺失或不匹配的标识符令牌来进行重新连接。

要避免这样，请确保在试图用相同的名称连接代理之前从 JON 库存删除被终止的代理，或者在启动新代理时指定正确的标识符令牌。

当代理被分配了不再匹配 JON 配置里记录的新的 VPN IP 地址时，您可能遇到另一个问题。例如主机被重启或 VPN 连接给终止时。在这种情况下，我们推荐您将 JON 代理的生命周期和 VPN 连接的生命周期绑定。如果连接掉线，您可以停止代理。当连接恢复时，请更新 `/etc/sysconfig/jon-agent-ec2` 里的 `JON_AGENT_ADDR` 来反映新的 IP 地址并重启代理。

关于如何修改代理的 IP 地址，请参考

https://access.redhat.com/site/documentation/JBoss_Operations_Network/ 上的《配置 JON 服务器和代理指南》。

如果需要启动和/或终止大量的实例，手动从 JON 库存里添加和删除就不太现实了。JON 的脚本功能可以用来自动化这些步骤。进一步的信息请参考 JON 文档。

[提交 bug 报告](#)

25.3.7. 配置实例来注册 JBoss Operations Network

使用下列过程来注册 JBoss EAP 6 实例到 JBoss Operations Network。

- 对于 JBoss EAP 6，在 User Data 字段里添加下列内容。

```
JON_SERVER_ADDR=jon2.it.example.com
## if instance not already configured to resolve its hostname
JON_AGENT_ADDR=`ip addr show dev eth0 primary to 0/0 | sed -n
's#.*inet \([0-9.]\+\)/.*#\1#p'`
PORTS_ALLOWED=16163
# insert other JON options when necessary.
```

请参考 第 25.4.1 节 “永久性的配置参数” 里以 **JON_** 开始的 JON 选项格式的参数。

[提交 bug 报告](#)

25.4. 用户脚本配置

25.4.1. 永久性的配置参数

概述

下面的参数可用来影响 JBoss EAP 6 的配置和操作。它们的内容会写入到 `/etc/sysconfig/jbossas` 和 `/etc/sysconfig/jon-agent-ec2` 里。

表 25.2. 可配置参数

实例类型	描述	默认值
JBOSS_JGROUPS_S3_PING_ACCESS_KEY	如果使用群集，用于 S3_PING 发现的 Amazon AWS 用户帐号的访问密钥。	N/A
JBOSS_JGROUPS_S3_PING_SECRET_ACCESS_KEY	Amazon AWS 用户帐号的秘密访问密钥。	N/A
JBOSS_JGROUPS_S3_PING_BUCKET	用于 S3_PING 发现的 Amazon S3 bucket。	N/A
JBOSS_CLUSTER_ID	群集成员节点的 ID。只用于群集。接受的值为（按顺序）： <ul style="list-style-type: none">● 0 - 1023 范围内的有效群集 DI 号码。● 网络接口名称，其值使用 IP 地址的最后八位。● “S3” 将通过 JGroup 的 S3_PING 使用的 S3 Bucket 协调 ID 的使用。 当所有群集节点都位于相同的 24 或更多位子网（例如，位于 VPC 子网）里时，我们推荐使用 IP 的最后八位（默认）。	eth0 地 IP 地址的最后八位

实例类型	描述	默认值
MOD_CLUSTER_PROXY_LIST	如果使用 <code>mod_cluster</code> ，用逗号隔开的 <code>mod_cluster</code> 代理的 IP/主机名的列表。	N/A
PORTS_ALLOWED	除了默认的端口，防火墙允许的转入端口列表。	N/A
JBOSSAS_ADMIN_PASSWORD	admin 用户的密码。	N/A
JON_SERVER_ADDR	要注册的 JON 服务器主机名或 IP。这仅用于注册，之后代理可能会与 JON 群集里其他服务器进行通讯。	N/A
JON_SERVER_PORT	代理与服务器通讯时使用的端口。	7080
JON_AGENT_NAME	JON 代理的名称，必须是唯一的。	实例的 ID。
JON_AGENT_PORT	代理侦听的端口。	16163
JON_AGENT_ADDR	JON 代理要绑定的 IP 地址。当服务器有多个公共地址（如 VPN）时使用它。	JON 代理默认会选择本地主机的 IP 地址。
JON_AGENT_OPTS	可用来配置 SSL、NAT 和其他高级设置的 JON 代理系统属性。	N/A

实例类型	描述	默认值
JBOSS_SERVER_CONFIG	<p>要使用的 JBoss EAP 6 服务器配置文件名称。如果 JBOSS_DOMAIN_CONTROLLER=true，那么 domain-ec2.xml 将被使用。否则：</p> <ul style="list-style-type: none"> 如果 S3 config 已存在，则使用 standalone-ec2-ha.xml。 如果指定了 MOD_CLUSTER_PROXY_LIST，那么将选择 standalone-mod_cluster-ec2-ha.xml。 如果没有使用这两个选项，那将使用 standalone.xml 文件。 也可以设置为 standalone-full.xml。 	standalone.xml 、 standalone-full.xml 、 standalone-ec2-ha.xml 、 standalone-mod_cluster-ec2-ha.xml 、 domain-ec2.xml ，取决于其他参数。
JAVA_OPTS	JBoss EAP 6 启动前添加到命令参数里的自定义值。	JAVA_OPTS 是有其他参数的值组成的。
JBOSS_IP	服务器要绑定的 IP 地址。	127.0.0.1
JBOSSCONF	启动的 JBoss EAP 6 配置集的名称。要阻止 JBoss EAP 6 启动，JBOSSCONF 应设置为 disabled 。	standalone
JBOSS_DOMAIN_CONTROLLER	设置这个实例是否将以域控制器运行。	false
JBOSS_DOMAIN_MASTER_ADDRESS	远程域控制器的 IP 地址。	N/A
JBOSS_HOST_NAME	逻辑主机名（域内）。这必须是可区分的名称。	HOSTNAME 环境变量的值。
JBOSS_HOST_USERNAME	当在域控制器里注册时主机控制器应使用的用户名。如果没有指定，将使用 JBOSS_HOST_NAME。	JBOSS_HOST_NAME
JBOSS_HOST_PASSWORD	当在域控制器里注册时主机控制器应使用的密码。	N/A

实例类型	描述	默认值
JBOSS_HOST_CONFIG	如果 JBOSS_DOMAIN_CONTROLLER=true，那么 host-master.xml 将被使用。如果 JBOSS_DOMAIN_MASTER_ADDRESSES 已存在，那么 host-slave.xml 将被使用。	host-master.xml 或 host-slave.xml ，取决于其他参数。
JBOSS_DOMAIN_S3_ACCESS_KEY	用于 S3 域控制器发现的 Amazon AWS 用户访问密钥。	N/A
JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY	用于 S3 域控制器发现的 Amazon AWS 用户秘密访问密钥。	N/A
JBOSS_DOMAIN_S3_BUCKET	用于 S3 域控制器发现的 Amazon S3 bucket。	N/A

[提交 bug 报告](#)

25.4.2. 自定义脚本参数

概述

User Data: 字段的用户定制部分可使用下面的参数。

表 25.3. 可配置参数

实例类型	描述
JBOSS_DEPLOY_DIR	活动的配置集的部署目录（例如， /var/lib/jbossas/standalone/deployments/ ）。放在这个目录里的可部署归档将被部署。Red Hat 推荐使用管理控制台或 CLI 工具而不是编辑配置文件来管理部署。
JBOSS_CONFIG_DIR	活动配置集的配置目录（例如： /var/lib/jbossas/standalone/configuration ）。
JBOSS_HOST_CONFIG	活动的主机配置文件的名称（例如， host-master.xml ）。Red Hat 推荐使用管理控制台或 CLI 工具而不是编辑配置文件来配置服务器。
JBOSS_SERVER_CONFIG	活动的服务器配置文件的名称（例如， standalone-ec2-ha.xml ）。Red Hat 推荐使用管理控制台或 CLI 工具而不是编辑配置文件来配置服务器。
USER_SCRIPT	自定义配置脚本的路径，在获取 user-data 配置前可用。
USER_CLI_COMMANDS	CLI 命令的自定义文件的路径，在获取 user-data 配置前可用。

[提交 bug 报告](#)

25.5. 故障排除

25.5.1. 关于 Amazon EC2 的故障解除

EC2 为每个实例提供了一个警告状态（Alarm Status），表示严重的故障。但缺乏这个警告并不保证实例已正确启动且服务运行正常。您可以使用 Amazon CloudWatch 及其自定义的度量功能来监控实例状态，但我们推荐使用企业级管理方案。

为了简化鼓掌接触，Red Hat 推荐用 JBoss Operations Network (JON) 管理 EC2 实例。通过安装的 JON 代理，它可以自动发现、监控和管理 EC2 实例上的许多服务，其中包括 JBoss EAP 6、Tomcat、Apache HTTP Server 和 PostgreSQL。关于用 JON 监控 JBoss EAP 的细节，请参考 [第 25.3.1 节“关于 AMI 监控”](#)。

[提交 bug 报告](#)

25.5.2. 诊断信息

在 JON、Amazon CloudWatch 或手动检测到问题时，诊断信息的常见来源是：

- `/var/log/jboss_user-data.out` 是 jboss-ec2-eap init 和用户自定义配置脚本的输出。
- `/var/cache/jboss-ec2-eap/` 包含实际的用户数据、启动实例使用的自定义脚本和自定义 CLI 命令。
- `/var/log` 也包含从主机启动、JBoss EAP 6、httpd 和其他大多数服务采集的日志。

访问这些文件只能通过 SSH 会话。关于如何配置和建立与 Amazon EC2 实例间的 SSH 会话，请参考《Amazon EC 起步指南》。

[提交 bug 报告](#)

附录 A. 补充的引用

A.1. 从红帽 CUSTOMER PORTAL 下载文件

必须具备的条件

- 在开始这个任务前，你需要一个 Customer Portal 帐号。访问 <https://access.redhat.com> 并点击右上角的 **Register** 链接来创建帐号。

过程 A.1. 登录红帽 Customer Portal 并下载文件

- 访问 <https://access.redhat.com> 并点击右上角的 **Log in** 链接。输入用户密码然后点击 **Log In**。

结果

你登录了 RHN 并位于主页位置 <https://access.redhat.com>。

- 进入 **Downloads** 页面。

使用下列选项来进入 **Downloads** 页面。

- 在顶部的导航栏里点击 **Downloads** 链接。
- 直接导航至 <https://access.redhat.com/downloads/>。

- 选择要下载的产品和版本。

使用下列途径来选择要下载的产品和版本。

- 逐级进行导航。
- 通过屏幕右边的搜索区来搜索产品。

- 根据操作系统和安装方法下载合适的文件。

根据你选择的产品，你可以选择 ZIP 归档、RPM、或特定操作系统和架构的原始安装程序。点击你要下载的文件右边的文件名或 **Download** 链接进行下载。

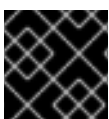
结果

文件将下载到你的电脑里。

[提交 bug 报告](#)

A.2. 在红帽企业版 LINUX 上配置默认的 JDK

在 Red Hat 企业版 Linux 上可以安装多个 JDK。这个任务向您展示如何指定当前环境使用的 JDK。它使用 **alternatives** 命令。



重要

这个任务只适用于 Red Hat 企业版 Linux。



注意

这个步骤也许不必要。Red Hat 企业版 Linux 将 OpenJDK 1.6 作为其默认选项。如果这就是您所要的，且您的系统运行正常，那您就不需要手动指定要使用的 JDK。

必须具备的条件

- 为了完成这个任务，您需要通过直接登录或者 `sudo` 命令具有超级用户权限。

过程 A.2. 配置默认的 JDK

1. 确定您首选的 `java` 和 `javac` 执行文件的路径。
您可以使用 `rpm -ql packagename |grep bin` 来查找从 RPM 安装的执行文件的位置。Red Hat 企业版 Linux 32 位系统上的 `java` 和 `javac` 执行文件的默认位置是：

表 A.1. `java` 和 `javac` 执行文件的默认位置

JDK	路径
OpenJDK 1.6	<code>/usr/lib/jvm/jre-1.6.0-openjdk/bin/java</code> <code>/usr/lib/jvm/java-1.6.0-openjdk/bin/javac</code>
Oracle JDK 1.6	<code>/usr/lib/jvm/jre-1.6.0-sun/bin/java</code> <code>/usr/lib/jvm/java-1.6.0-sun/bin/javac</code>

2. 设置您要使用的替代 JDK。
运行下列命令设置您的系统来使用专有的 `java` 和 `javac`: `/usr/sbin/alternatives --config java` 或 `/usr/sbin/alternatives --config javac`。请遵循屏幕上的说明进行。
3. 可选项：设置 `java_sdk_1.6.0` 的替代选项。
如果您想使用 Oracle JDK，你也需要为 `java_sdk_1.6.0` 设置替代选项。请使用下列命令：`/usr/sbin/alternatives --config java_sdk_1.6.0`。正确的路径通常是 `/usr/lib/jvm/java-1.6.0-sun`。

结果：

已选择并激活了替代 JDK。

[提交 bug 报告](#)

附录 B. 修订记录

修订 6.3.0-38

Monday August 4 2014

Sande Gilda

Red Hat JBoss 企业级应用程序平台 6.3.0.GA